SmartDOTS - A Framework for Efficient Data Synchronization on Mobile Devices

Werner Kurschl, Stefan Mitsch, Rene Prokop
Upper Austrian University of Applied Sciences - Research and Development Competence Center
Hauptstraße 117, A-4232 Hagenberg, Austria

Abstract

Mobile enterprise applications typically access data from the enterprise's various applications. Allowing the mobile application to access these data online only would be a major hindrance for mobile workers that cannot assume a constantly available network connection. We present a middleware called Smart Data Off The Spot (SmartDOTS) for building nomadic distributed enterprise applications. In SmartDOTS, the business data model is represented per device, network, and task; this representation configures SmartDOTS to provide business data tailored to the specific capabilities (e.g., memory capacity) of a device, and needs of a task. The SmartDOTS approach is independent of persistent storages; it is part of the application developer's task to provide server-side persistency. Thus, any (legacy) application can be integrated with SmartDOTS.

1. Introduction

When developing distributed enterprise applications, developers usually do not explicitly handle problems related to distributed data access. Middleware provides a higher level of abstraction and hides the complexity introduced by distribution.

Current middleware for distributed systems typically reach for complete transparency (see [2]); their main goal is to hide heterogeneity (e.g., the device's location and access to networked resources) to make remote resources appear as locally available. This approach is successfully used for stationary distributed systems with fixed networks and thus constant network connectivity and quality of service. But it seems not to be suitable for mobile devices (see [10]) for the following reasons: distributed transactions, object requests or remote procedure calls assume a connection that is constantly available, but in mobile systems, varying network environments with typically sporadic connectivity and low bandwidth are the norm. Furthermore, most middleware support synchronous communication that requires the client and server to be up and running simultaneously, but in a mobile environment the client and server are often not connected at the same time (e.g., due to no network coverage, or no network connection to save battery power).

Depending on the intended application scenario, middleware needs to be tailored to meet the distinct challenges of the scenario. SmartDOTS is designed for nomadic distributed systems as they are often met in industrial environments, see Fig. 1.

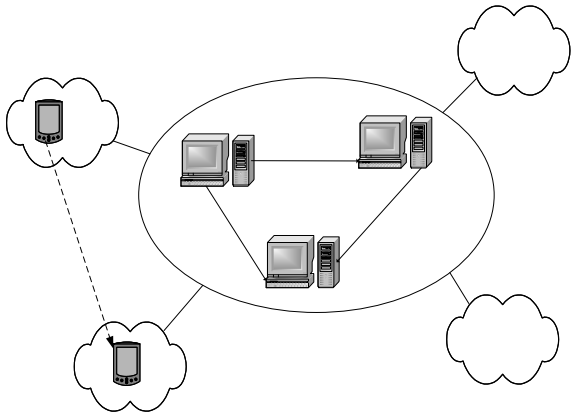

Figure 1. Structure of a nomadic distributed system (cf. [10]).

A nomadic distributed system is based on a core of fixed hosts; at the periphery wireless access points provide access to this fixed network for mobile devices.

As proposed in [10], SmartDOTS follows a semi-transparent approach that does not hide all connection issues from the application. Hence, an application built upon SmartDOTS can adapt to different connection conditions as it can typically make more efficient and better-quality decisions based on application-specific information than SmartDOTS itself. It can, e.g., rely on business logic or ask the

user for decisions in ambiguous situations.

Mobile enterprise applications (like MOSES [8], a mobile safety system for work clearance processes) often support collaborative working processes. The mobile work force needs to stay in contact with a central office; however, networks are unreliable and a connection might not always be available or it can be slow. Thus, the mobile workers must also have access to enterprise data when being offline. Data needs to be replicated to the mobile device—within the constraints of available local memory—for offline access; changes and updates to these data need to be tracked and synchronized with the enterprise applications when a network connection can be reestablished.

Data replication can be performed using some form of predictive algorithms (see [1]). But we believe this solution has a major drawback: predictions can be wrong. We promote a less general solution that treats data replication as part of the business logic. SmartDOTS provides the necessary support for replicating data based on a application developer or user-defined synchronization model.

Enterprises typically comprise several applications that might be custom built, acquired, or part of a legacy system (see [7]). Mobile enterprise applications often integrate more than one application into a single mobile application and thus rely on data provided by each of these. Integrating data from multiple applications usually increases the bandwidth requirements for data synchronization. Additionally, these applications often show high latency, but users need a fluent work experience; they do not want to be hindered in their daily work. It is therefore important to provide an efficient and predictable replication and synchronization mechanism for mobile software applications.

Besides the described technical needs for providing data offline on the device, economical reasons demand for it too. Wireless networks are often cost intensive; typically, faster networks are more expensive then slower ones. Additionally, extensive network access imposes an increased energy drain on the batteries (see [11]). In industrial environments, where workers frequently need to work in the field for a whole shift (i.e., eight hours) energy consumption needs to be kept at a minimum.

2. Related Work

XMIDDLE is a mobile middleware for transparent sharing of XML documents in a peer-to-peer network (see [9]). XMIDDLE supports sharing of tree-structured data between peers; therefore, each peer offers access points for other peers to manipulate its data online or to replicate the data for working offline.

Bayou ([3]) is a platform of replicated mobile databases (hosted on mobile devices) on which to build collaborative applications. A common scenario are several users sharing data while being disconnected from the rest of the system. SmartDOTS differs from Bayou in accessing, replicating and synchronizing any data from enterprise applications.

SodaSync, described in [1], is a programming framework that provides a generic synchronization model for mobile enterprise applications. It can integrate multiple heterogeneous backend data stores through a unified high-level data model. SodaSync focuses on the interface between an idealized generic data model and real data sources. Data is modeled using a Service Data Object (SDO)-based representation of application data; this approach limits SodaSync to Java applications based on the SDO framework. Furthermore, it leaves important aspects for mobile applications like device-, network and task-specific data replication, and context-dependent network selection unsolved.

The SyncML standard (see [6]) is a specification for an interoperable data synchronization framework using an XML-based format. SyncML can be used as an underlying protocol for data exchange in SmartDOTS.

In contrast to the described systems that primarily synchronize single data stores, SmartDOTS focuses on synchronization of data for complete applications that integrate multiple data stores. Additionally, SmartDOTS allows application designers to specify data packages for synchronization. These data packages can be tailored to fit tasks, devices, and networks; they usually describe a set of data needed to perform a specific task on a specific device and are designed for “take-away”. SmartDOTS focuses on providing synchronization capabilities over real world (often synchronous) communication networks and models, and over multiple data stores. SmartDOTS aims to provide this functionality as a service to application developers.

3. The SmartDOTS Approach

To overcome some of the above mentioned shortcomings we developed the SmartDOTS (Smart Data Off The Spot) approach. This approach is driven by the observation, that mobile industrial workers typically need a known set of data to perform a specific task. The devices used in an industrial environment often are of diverse nature; notebooks, cell phones, and PDAs are used interchangeably. Mobile workers often start their work at a location with constant network connection (e.g., the office), but in the course of their task they stay in areas with only sporadic connectivity. Thus, data can be packaged to fit the task at hand and loaded to the device while a network connection is available. Then this data can be worked on even without a network connection. The tasks of mobile industrial workers are often collaborative. Hence, synchronization of data is performed whenever a network connection is available.

The core of SmartsDOTS is a synchronization model that describes data packages for ”take-away” that fit specific

tasks, devices, and networks. The synchronization model acts as a configuration to SmartDOTS. The packages are prefabricated to remove the sensitivity to latency exposed by legacy applications. The enterprise data is exactly tailored to a task, device, and network. A powerful device (like a notebook) can replicate more detailed data for several tasks, while a less capable device (a PDA or mobile phone) gets less detailed data or might concentrate on a single task. The available network connection also influences selecting a package for replication. Even a powerful device on a slow and low-bandwidth connection will prefer to replicate a small package with less detailed data.

The data in a package is independent of the data representation in legacy systems. SmartDOTS operates on the object-oriented data model provided by any server business-logic that integrates legacy systems.

Since a mobile worker cannot assume a constantly available network connection, SmartDOTS communicates its current state to its users. Mobile enterprise applications (and especially the end users) are often interested whether data is provided from the enterprise directly (online) or from a previously synchronized local data source (offline). Furthermore, they also get informed whether data collected offline is already synchronized to the enterprise.

4. The Synchronization Model

The synchronization model describes devices, networks, costs, and packages of business data. A package contains business data that is needed for a specific task (typically a small fraction of the complete business data contained in an enterprise application) and tailored to fit a specific device and network. The business data to include in a package is represented by its data types and the references between those types. We call this representation *business data model*. A reference can either be *weak* or *strong*. When building packages, strong references are resolved (i.e., the referenced data gets added to the package); weak references are not resolved, but the referenced data is loaded on demand. A package defines a query to retrieve the root objects. Following the references from the root objects builds the data contained in a package.

The synchronization-model consists of the following parts: *(i)* package, *(ii)* device, *(iii)* network, *(iv)* cost model, *(v)* type, and *(vi)* reference.

A *package* describes data packed by SmartDOTS for immediate retrieval by mobile clients. It contains business data that is needed for a specific task and tailored to fit a specific device and network.

A *device* describes the capabilities, like storage capacity, processing power, and supported networks, of a specific class of devices.

The *network* describes a network by its maximum and average transmission rate and it contains a cost model for the data to be transferred.

The *cost model* describes the price for transferring data over a network at a specific point in time. Network providers often charge data transfer differently during peak (business) hours and off-peak hours. The cost model allows the SmartDOTS-Service to decide, which network to use for sending a package. Packages with a low priority are only sent over cheap or free-of-charge networks, while packages with high priority are also allowed to be sent over costly networks.

Types and *references* describe the business data contained in a package. Typically, not the complete business data contained in an enterprise application is to be added to a package. Data can be limited through the following two measures: *(i)* not the complete object graph gets added to a package, and *(ii)* an object's properties and values are only partially added. Strong references between types indicate data that has to be kept together in a package, while weak references are resolved on demand during the work. A type can be restricted to add only parts of its properties to a specific package.

5. Architecture

This section describes the design of SmartDOTS. SmartDOTS, depicted in Fig. 2, consists of a server-side *SmartDOTS-Service* and a client-side *SmartDOTS-Engine*.

The SmartDOTS-Service integrates the enterprise application's server business logic. The server business logic typically accesses data in different data sources (often legacy systems) and combines it in a single object-oriented business data model. The SmartDOTS-Service continuously builds packages containing data from this business model and stores them for immediate retrieval in a *package cache*. Thus, sensitivity to the legacy systems' latency is eliminated. The content of the packages can be configured on a device, network, cost, etc. basis (as described above in the synchronization model).

The SmartDOTS-Engine is the primary data storage for an application. The application can query the SmartDOTS-Engine for data; SmartDOTS provides these data in the application's business model, similar to directly accessing the server business logic. The SmartDOTS-Engine retrieves data in packages (described by the synchronization model) from the SmartDOTS-Service and stores it in a local replica. The communication between the mobile client application and the application's server business logic is completely asynchronous (even over a synchronous connection). Yet, through the local replica the SmartDOTS-Engine is able to offer a synchronous programming model to its clients, which is more familiar and convenient to most programmers than asynchronous or message oriented programming.

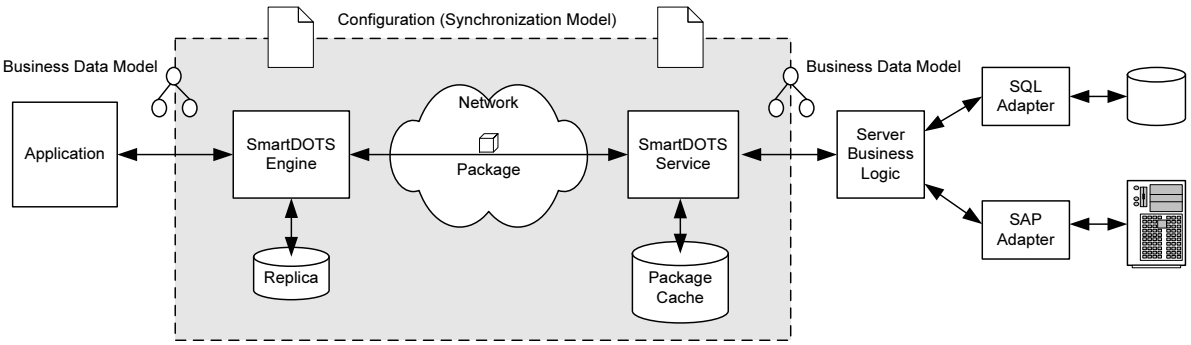

Figure 2. The SmartDOTS architecture.

The SmartDOTS-Engine handles data replication, tracks changes and updates to local data, and synchronizes the local replica with the enterprise applications.

Communication between the SmartDOTS-Engine and the SmartDOTS-Service is not limited to a particular data representation or communication channel, though, for performance reasons, binary protocols are preferred.

6. The SmartDOTS-Service

The SmartDOTS-Service, shown in Fig. 3, integrates an enterprise application through the interface *IBusiness-Model*. The Packer prefabricates packages from the enterprise application's data and stores them in the package cache for later retrieval through the SmartDOTS-Engine.

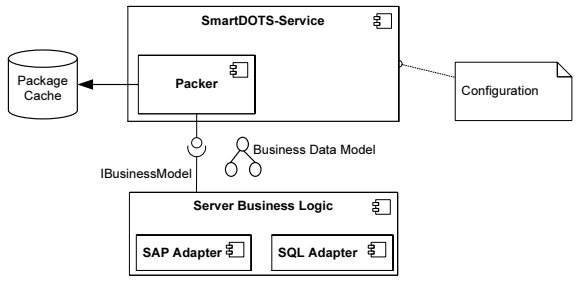

Figure 3. The SmartDOTS-Service.

The SmartDOTS-Service (including the Packer) is configurable through the synchronization model described above; it defines which packages to build, which data the packages have to contain, and when the package cache has to up-dated.

The SmartDOTS-Service can be queried for packages. A *package query* contains the device, the network, and the user/task; the package cache is then searched for packages matching the query. Additionally, a matching synchronization model is provided to configure the SmartDOTS-Engine on the client device (e.g., which network to use for synchronizing data).

7. The SmartDOTS-Engine

The SmartDOTS-Engine offers features to persist and synchronize a business model with the SmartDOTS-Service. Therefore it has to track accesses to the SmartDOTS-Service, handle data synchronization and perform automatic service-relocation when the service is not reachable.

To implement this functionality we identified three basic approaches: *(i)* The client-side business model contains, additionally to the current data, an original state. Thus, data changes can be collected from the business model itself. This approach is similar to the implementation of ADO.NET Datasets. *(ii)* All activities on the business model are tracked by the SmartDOTS-Engine and written into a log that describes the delta between the local data and the server data. *(iii)* Neither the business model holds changes, nor activities on the data are tracked. Instead, changes are determined by comparing the current data with a persisted original state, which could be stored in the local replica.

We chose the second approach, which is described subsequently. Note that the concepts of automatic service-relocation, data replication and synchronization apply to all three approaches with only minor changes.

7.1. Data Management, Replication and Synchronization

To enable an unhindered working process, it's necessary to decouple the availability of data from the network conditions. This requires a local data management on the mobile device, which fetches data from the server, holds it locally, and keeps it up to date on both sides.

SmartDOTS distinguishes between data replication prior to working in the field and data synchronization while working mobile. Data replication is the initial step of duplicating the enterprise application's data to the mobile device. This process can be triggered by the user login, which automatically leads to download of the user's configuration and initial data package. Alternatively, the user can activate the replication by selecting a new task which makes a new data package necessary. Usually, this is done when a fast and reliable network connection is available, as typically a large amount of data needs to be replicated. In contrast, data synchronization is handled by SmartDOTS automatically during the working process to reconcile the mobile database with the enterprise application's databases.

Data replication can be simplified compared to data synchronization, as during replication conflicting updates are assumed not to happen. Thus, data replication always overwrites data on the mobile device with data from the enterprise applications. Data replication then is reduced to the problem of minimizing the amount of data (and with it the needed time) for transmission. Trachtenberg et al. in [13] propose fast synchronization using characteristic polynomial interpolation.

Because of the different requirements of the mobile devices SmartDOTS' data management offers an abstraction of storage media—named *DataStore*—and is able to work with several DataStores concurrently, independent of their location. SmartDOTS uses a configuration to arrange these DataStores. A mobile device, which is prepared for working disconnected, typically uses *(i)* a local DataStore that holds the data in a local replica, *(ii)* a remote DataStore that is connected to the SmartDOTS-Service, and *(iii)* a logging DataStore that tracks the changes on the local data. The usage of these DataStores depends on the current network state and can be influenced by the automatic service-relocation (see Sect. 7.2): as long as the mobile device is online, the changes are done on the local data (local DataStore) and submitted immediately to the server (remote DataStore). Therefore the changes are performed on the local DataStore and the remote DataStore. When the mobile device becomes disconnected, the remote DataStore is not reachable any more and therefore all changes are performed on the local DataStore and on the logging DataStore. This logging DataStore tracks the changes and builds up a description of the delta between the local data and the data on the server.

Synchronization is performed upon service-relocation from offline access of the local replica to online access of the server. By replaying the delta, which was tracked by the logging DataStore, a synchronous state can be reestablished on the remote DataStore.

The management of replicated and distributed databases requires algorithms for guaranteeing data consistency and for resolving conflicting updates. Several architectures, such as Bayou [12] and SodaSync [1], have been proposed to address these important problems. SmartDOTS was in the first stage designed for an environment where conflicting updates cannot occur. We consider resolving conflicting updates as part of our future work.

7.2. Automatic Service-Relocation

When no network connection is available, the SmartDOTS-Engine performs an automatic service-relocation to provide data either using a different connection (e.g., GPRS instead of WLAN) or from the local replica.

We use a combination of the *Bridge* and *State* pattern, both described in [5], to enable the service-relocation (see Fig. 4).

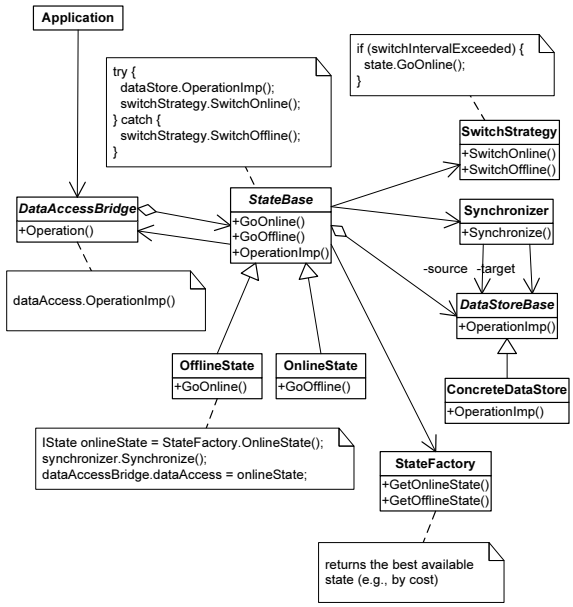

Figure 4. Combined Bridge and State pattern.

The Bridge hides the concrete State (online or offline) currently in use from its clients. Incoming requests to process a transaction are forwarded to the current State, which in turn forwards it to its DataStore (e.g., the SmartDOTS-Service or the local replica). When the DataStore is not able to process the request (e.g., because the network connection fails), the State asks the switch strategy to switch offline. The *StateFactory* provides the State to switch to (e.g., depending on the costs involved with using the State). The Bridge is then modified to access the *OfflineState* in subsequent requests; the current request is satisfied by the OfflineState as well. Upon successful completion of a request, a State asks the *SwitchStrategy* to switch online. The

SwitchStrategy decides to switch online, if e.g., a network connection is available and a defined switching interval is expired.

7.3. Transactions

In the second approach, SmartDOTS tracks all activities on the business data model. Atomic units of work are reported to SmartDOTS and represented by transactions. Our transaction applies Fowler's pattern *Unit of Work* (see [4]). It keeps track of everything that is done during a business transaction and affects the enterprise data (e.g., objects being created, modified or deleted). When the transaction is committed, it gets executed on the DataStores. Each DataStore can decide what to do with a transaction, but usually it leads to a package being sent to the SmartDOTS-Service or tracked locally. A transaction is always executed successfully on all DataStores or it gets rolled back.

A transaction represents one or more actions to perform. These actions usually have results that are collected by the transaction and provided to the client.

8. Conclusion and Further Work

We present the SmartDOTS framework for building nomadic distributed enterprise applications that supports data replication and synchronization from various enterprise applications. The framework allows the modeling of business data, so that device-, network and task-specific data replication, and context-dependent network selection can be modeled and automatically be executed by the framework. It solves the problem of using a heterogeneous set of mobile devices with different capabilities by modeling and configuration techniques.

We implemented a mobile enterprise application named MOSES based on a partial implementation of SmartDOTS. MOSES currently works on PDAs and notebooks. As part of our future work, we will enhance data synchronization; especially we will focus on resolving conflicting updates. We will also extend MOSES to support smaller devices (smart-phones or cell-phones) to further evaluate SmartDOTS with very limited devices.

References

[1] P. Castro, F. Giraud, R. Konuru, A. Purakayastha, and D. Yeh. A Programming Framework for Mobilizing Enterprise Applications. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, pages 196–205, Washington DC, USA, 2004. IEEE Computer Society.

[2] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems—Concepts and Design*. Addison-Wesley, Boston, 3rd edition, 2001.

[3] A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and B. B. Welch. The Bayou Architecture: Support for Data Sharing Among Mobile Users. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, USA, 1994. IEEE.

[4] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, 2003.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, 1995.

[6] U. Hansmann, R. Mettala, A. Purakayastha, and P. Thompson. *SyncML: Synchronizing and Managing Your Mobile Data*. Prentice Hall, 2002.

[7] G. Hohpe and B. Woolf. *Enterprise Integration Patterns*. Addison-Wesley, Boston, 2004.

[8] W. Kurschl, S. Schmid, and C. Domscha. MOSES - A Mobile Safety System for Work Clearance Processes. In *Proceedings of the 4th International Conference on Mobile Business*, pages 166–172, Sydney, Australia, 2005. IEEE.

[9] C. Mascolo, L. Capra, and W. Emmerich. An XML Based Middleware for Peer-to-Peer Computing. In *Proceedings of the International Conference on Peer-to-Peer Computing*, pages 69–74. IEEE Computer Society, 2001.

[10] C. Mascolo, L. Capra, and W. Emmerich. Middleware for Mobile Computing (A Survey). In *Networking 2002 Tutorial Papers*, pages 20–58. Springer, 2002.

[11] T. Starner. Powerful change part 1: Batteries and possible alternatives for the mobile market. *IEEE Pervasive Computing*, 2(4):86–88, 2003.

[12] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP)*, pages 172–183, Copper Mountain Resort, USA, 1995. ACM.

[13] A. Trachtenberg, D. Starobinski, and S. Agarwal. Fast PDA Synchronization Using Characteristic Polynomial Interpolation. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Computer Societies (INFOCOM)*, New York, USA, 2002. IEEE.