An XML based Middleware for Peer-to-Peer Computing

Cecilia Mascolo, Licia Capra, and Wolfgang Emmerich
Dept. of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{C.Mascolo|L.Capra|W.Emmerich}@cs.ucl.ac.uk

Abstract

An increasing number of distributed applications will be written for mobile hosts, such as laptop computers, third generation mobile phones, personal digital assistants, watches and the like, with focus on peer-to-peer collaboration. Application engineers have to deal with a new set of problems caused by mobility, such as low bandwidth, context changes or loss of connectivity. During disconnection, independently from each others, users will typically update local replicas of shared data, possibly generated by peers. The resulting inconsistent replicas need to be reconciled upon re-connection. To support building mobile applications that use both replication and reconciliation over ad-hoc networks, we have designed XMIDDLE*, a peer-to-peer middleware that targets mobile computing settings. In this paper we describe* XMIDDLE *and give a flavour of how reflection capabilities are used to allow application engineers to influence replication and reconciliation techniques.* XMIDDLE *enables the transparent sharing of XML documents across heterogeneous mobile peers, allowing on-line and off-line access to data.*

1 Introduction

The spread of mobile computing devices in the recent years has been very fast. Mobile phones become increasingly computationally powerful, are integrated with PDA capabilities (e.g., Nokia's 9210) and are equipped with ad-hoc networking technologies (e.g., Ericsson's T36 that implements Bluetooth). These enable new classes of applications to exploit, for example, the ability to form ad-hoc workgroups and share peer resources; but they also present new challenges to the mobile application developer. In particular, resources, such as available main memory, persistent storage, CPU speed and battery power are scarce and need to be exploited efficiently. Moreover, network connectivity may be interrupted instantaneously and network bandwidth will remain by orders of magnitude lower than in wired networks.

In distributed systems, the complexity introduced through distribution is made transparent to the application programmer by means of middleware technologies, which raise the level of abstraction. Existing middleware technologies, such as remote procedure call systems, distributed object middleware, and message- or transaction-oriented systems hide the complexities of distribution and heterogeneity from application programmers and thus support them in constructing and maintaining applications efficiently and cost-effectively. However, these technologies have been built for wired networks and are unsuitable for a mobile setting [3, 9]. In particular, the interaction primitives, such as remote procedure calls, object requests, remote method invocations or distributed transactions that are supported by current middleware paradigms assume a high-bandwidth connection of the components, as well as their constant availability. In mobile systems, instead, unreachability and low bandwidth are the norm rather than a failure. In Bayou [13] disconnection was contemplated as a rare and occasional event. The system hides mobility from the application layer in the same way as transparency for relocation of object is used in modern middleware systems.

We rather believe that middleware systems for mobile computing need to find different kinds of interaction primitives to accommodate the possibility for mobile components to become unreachable. Many PDA applications copy, for example, agendas, to-do lists and address records from a desktop machine into their local memory so that they can be accessed when the desktop is unreachable. In general, mobile applications must be able to replicate information in order to access them off-line. Replication causes the need for synchronization when a connection is re-established. This need is not properly addressed by existing middleware systems. The commonly used principle of transparency prevents the middleware to exploit knowledge that only the application has, such as which portion of data to replicate and which reconciliation policy to apply. It seems therefore necessary to design a new generation of middleware systems, which disclose information previously hidden, in order to make best use of the resources available, such as local memory and network bandwidth.

Tuple space coordination primitives, have been employed to facilitate component interaction for mobile systems. Tuple spaces achieve a decoupling between interacting components in both time and space by matching the idea of asynchronicity with the mobile computing embedded concept of disconnection and reconnection. Tuple spaces do not impose any data structures for coordination allowing more flexibility in the range of data that can be handled. On the other hand the lack of any data structuring primitives complicates the construction of applications that need to exchange highly structured data.

Peer-to-peer systems [12] have been usually developed for fixed infrastructures. However, their rich paradigms for resource and data sharing among peers can be naturally extended to mobile settings. In this paper we present XMIDDLE, which advances mobile computing middleware approaches by firstly choosing a more powerful underlying data structure and secondly by supporting application-driven replication and reconciliation. XMIDDLE's data structure are trees rather than tuple spaces. XMIDDLE allows peer-to-peer sharing of data trees and off line manipulation of peers information. More precisely, XMIDDLE uses the eXtended Markup Language (XML) to represent information and uses XML standards, most notably the Document Object Model (DOM) to support the manipulation of its data. This means that XMIDDLE data can be represented in a hierarchical structure rather than, for instance, in a flat tuple space. The structure is typed and the types are defined in an XML Document Type Definition or XML Schema. XMIDDLE applications use XML Parsers to validate that the tree structures actually conform to these types. The introduction of hierarchies also facilitates the coordination between mobile hosts at different levels of granularity as XMIDDLE supports sharing of subtrees. Furthermore, representing mobile data structures in XML enables seamless integration of XMIDDLE applications with the Micro Browsers, such as WML browsers in mobile phones, that future mobile hosts will include.

The paper is organized as follows: in Section 2 we briefly introduce XMIDDLE and the main characteristics of the system. XMIDDLE makes extensive use of XML and we sketch how we deploy XML and related technologies in Section 3. Section 4 discusses the basic architecture of XMIDDLE and presents the primitives that this architecture provides for mobile application. In Section 5 we discuss and evaluate the XMIDDLE system and in Section 6 we conclude the paper and list some future work.

2 An Outline of XMIDDLE

XMIDDLE allows hosts (i.e., PDAs, mobile phones, laptop computers or other wireless devices) to be physically mobile, while yet communicating and sharing information with other hosts. We do not assume the existence of any fixed network infrastructure underneath. Mobile peers may come and go, allowing complicated *ad-hoc network* configurations. Connection is symmetric but not transitive as it depends on distance; for instance host $\mathtt{H}_A$ can be connected to host $\mathtt{H}_B$, which is also connected to host $\mathtt{H}_C$. However, host $\mathtt{H}_A$ and host $\mathtt{H}_C$ may be not connected to each other. Mobile network technologies, such as Bluetooth facilitate these configurations with multiple so called *piconets* whose integration forms *scatternets* in Bluetooth.

In order to allow mobile devices to store their data in a structured and useful way we assume that each device stores its data in a tree structure. Trees allow sophisticated manipulations due to the different node levels, hierarchy among the nodes, and the relationships among the different elements which could be defined. XMIDDLE defines a set of primitives for tree manipulation, which applications can use to access and modify the data.

When hosts get in touch with each other they need to be able to communicate. XMIDDLE therefore provides an approach to sharing that allows on-line collaboration, off-line data manipulation, synchronization and application dependent reconciliation. On each device, a set of possible access points for the private tree are defined so that other devices can link to these points to gain access to this information; essentially, the access points address branches of trees that can be modified and read by peers. In order to share data, a host needs to explicitly *link* to another host's tree. The concept of linking to a tree is similar to the mounting of network file systems in distributed operating systems to access and update information on a remote disk. Access points to a host's tree are a set that we call $ExportLink$. XMIDDLE allows mobile hosts to share data when they are connected or replicate the data and perform operations on them off-line; reconciliation of data takes place once the hosts reconnect. A host also records the branches that it links from other remote hosts in the set $LinkedFrom$, and the hosts linking to branches of the owned tree in the set $LinkedBy$. These sets contain lists of tuples $(host, branch)$ that define the host that is linking to a branch, and from whom a branch is linked, respectively. $LinkedFrom$ does not mirror the connection configuration, that is, host H_A can be in the $LinkedFrom$ list of H_B also if the two hosts are not in reach (specific primitives for $linking$ and $unlinking$ trees modify these sets). On the contrary, the $LinkedBy$ set is updated by *connection* and *disconnection* operations and it is used to know to whom to notify changes of parts of the tree.

The link operation is, however, not enough for data sharing among mobile hosts; in order to share data, hosts needs to be *connected*. A host $\mathtt{H}_A$ becomes $connected$ with another host $\mathtt{H}_B$ when it is “in reach”. When two hosts are connected they can share and modify the information on each other’s linked data trees. Each host has full control over its own tree, however it is obliged to notify other connected hosts that link to the modified part ($branch$) of its tree (i.e., all the H_i which are in its $inReach$ set and listed in the $LinkedBy$ set as linking the modified $branch$) about the changes introduced. If H_i wishes to modify a $branch$ linked from a remote host H_j which is in reach, it requests H_j to perform the desired changes. H_j then notifies the changes to all the hosts (in reach) that link to the modified branch, including H_i.

Hosts may explicitly disconnect from other hosts using the $disconnect$ primitive, even though these hosts may be “in reach”. XMIDDLE supports explicit disconnection to enable, for instance, a host to save battery power, to perform changes in isolation from other hosts and to not receive updates that other hosts broadcast. Disconnection may also occur due to movement of a host into an out of reach area, or to a fault. In both cases, the disconnected host retains replicas of the last version of the trees it was sharing with other hosts while connected and continues to be able to access and modify the data; a versioning system is in place to allow consistent sharing and data reconciliation (more details in [11]).

3 XMIDDLE and XML

In the previous section we have described the motivation and main characteristics of XMIDDLE. We now give the details on how we use XML for structuring the device information as trees, and how the XML related technologies are exploited in order to achieve linking and addressing.

XML documents can be semantically associated to trees. We therefore format the data located on the mobile devices as XML trees. The applications on the devices are enabled to manipulate the XML information through the DOM (Document Object Model) [1] API which provides primitives for traversing, adding and deleting nodes to an XML tree. The implementation of this API, however, is XMIDDLE specific.

Furthermore, XML related technologies such as XPath [4] and XLink [10] are used in XMIDDLE to format the linking and addressing of points in a tree and reference remote trees. $LinkedFrom$, $LinkedBy$ and $ExportLink$ sets are formatted using the XPath and XLink syntax. The XPath syntax is very similar to the Unix directory addressing notation. For instance, to address a node in an XML tree the notation used is `/root/child1/child2`. XLink builds on top of XPath; it allows the addressing of specific nodes in remote XML documents without affecting the documents at all (the links are stored in a separate document called “linkbase”). We use XLink technology to define the links between devices and remote tree entry points.

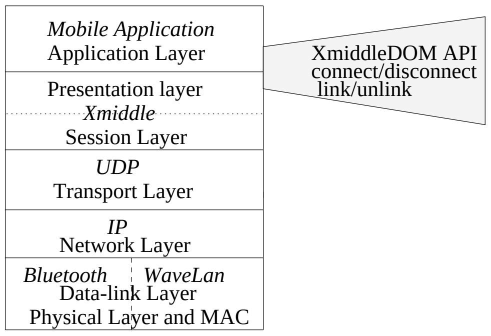

Figure 1. The protocol stack for mobile environments using XMIDDLE.

The reconciliation of XML tree replicas which hosts use to concurrently and off-line modify the shared data, exploits the tree differencing techniques developed by IBM. IBM XMLTreeDiff is a package that implements this algorithm and that XMIDDLE uses to handle reconciliation. We note, however, that reconciliation cannot in all cases be completed by the XMIDDLE layer alone. Similarly to merging text files, tree updates may lead to differences which can be solved only using application-specific policies or may even need end-user interaction. The use of XML as an underlying data structure, however, enables XMIDDLE to both highlight the differences and define reconciliation policies specific to particular types of document elements, and therefore to specific applications (more details in [11]).

4 The XMIDDLE Architecture

We now present an overview of the XMIDDLE architecture, which follows the ISO/OSI reference model. XMIDDLE implements the session and presentation layers on top of standard network protocols, such as UDP or TCP, that are provided in mobile networks on top of, for instance, a Bluetooth data-link layer (i.e., Logical Link Control and Adaptation Protocol) and MAC and physical layer (i.e., Bluetooth core which is based on radio communication). Our current prototype is however based on UDP upon Wireless Lan, which is an other possible option. The protocol stack for XMIDDLE is shown in Figure 1.

The presentation layer implementation maps XML documents to DOM trees and provides the mobile application layer with the primitives to link, unlink and manipulate its own DOM tree, as well as replicas of remote trees. The session layer implementation manages connection and dis-

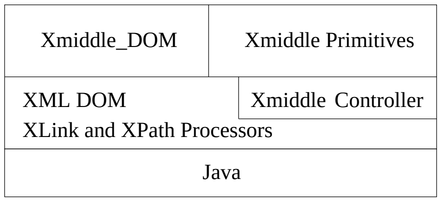

Figure 2. The XMIDDLE architecture.

connection.

Figure 2 refines the presentation and session layer implementations of XMIDDLE. The *Xmiddle Controller* is a concurrent thread that communicates with the underlying network protocol and handles new connections and disconnections, triggers the reconciliation procedures and handles reconciliation conflicts according to application specific policies. As XMIDDLE is entirely implemented in Java, it relies on a Java Virtual Machine (JVM). A large variety of JVMs have been implemented for mobile devices. The Symbian operating system for the third generation of mobile phones, for example, has a Java Virtual Machine built in. Likewise, Sun provides a minimal kernel virtual machine (KVM) implementation for Palm PDAs.

The *Xmiddle Primitives* API provides mobile applications with operations implementing the XMIDDLE primitives, such as link, unlink, connect and disconnect. The ability to link to trees from other devices introduces a client/server dependency between mobile hosts. We refer to the host which a tree is linked from as the *server host* and the host that links the tree as a *client host*. The XMIDDLE implementation maintains this client/server relationship in the `LinkedFrom` and `LinkedBy` tables that are kept on each host (they correspond to the sets with the same names defined in Section 2). The `LinkedFrom` table also needs to keep track of the host that owns a subtree in order to allow the application to be able to request updates from that host; this is done using XLink. It is also necessary to the hosts that have linked to a tree for being able to broadcast updates when the hosts are in reach.

The *Xmiddle_DOM* component provides the XMIDDLE implementation of the DOM to mobile applications.

5 Discussion and Related Work

We have described XMIDDLE and shown its architecture. Synchronization and data locking have been described as main problems in wireless environments by Imielinski and Badrinath in [8]. XMIDDLE offers a possible solution.

We focus our interest on ad-hoc networks where host configurations are relative and dynamic. No discovery services are set-up as in Jini as all the hosts have the same capabilities. They are able to reconfigure their own connection groups while they move, through connection and disconnection with the other hosts.

Tuple space based systems for logical and physical mobility such as JavaSpaces [6], Lime [14], TSpaces [7], and Mars [2] exploit the decoupling in time and space of these data structures in the mobility context where connect and disconnect are very relevant and frequent operations. However, tuple spaces are very general and loose data structures, which do not allow complex data organizations and therefore do not fit all the application domains. XML allows us to introduce hierarchy of data and to address specific paths in the structure so that more elaborated operations can be performed by the applications.

An additional disadvantage of tuple-space based systems is in term of synchronization capabilities. Tuple-spaces are multi-sets, which means every tuple can be duplicated in the space. Whenever two or more devices, which replicate a piece of data (represented as a tuple), disconnect and modify it the reconciliation process of rejoining the tuple spaces during reconnection becomes an unnatural operation (due to the multi-set property of tuple spaces).

The issue of data replication and synchronization has been addressed in the context of distributed file systems by Coda, which adopts an application-transparent adaptation technique, and its successor Odyssey[15], which enables application-aware adaptation. Compared to these approaches, XMIDDLE firstly defines a different level of granularity of the data that can be moved across mobile devices, that is, parts of an XML document, as small as we wish, as opposed to whole files. This may have a relevant impact when dealing with slow and/or expensive connection. Moreover, we do not assume the existence of any server that is more capable and trustworthy than mobile clients, as we target pure ad-hoc network configurations. Finally, the use of XML adds semantic to the replicated data, against the uninterpreted byte streams of files; this added semantics can then be exploited to provide better conflict detection and resolution policies from an application point of view.

XMIDDLE uses only XML trees as data structures and exploits the power of the nature of the data structure with specific operations; for instance, the linking primitive facilitates off-line sharing of information, which is very valuable in mobile computing contexts where hosts have the need to move away from the source of information even if they may want to continue to work on the downloaded data. Reconciliation mechanisms are needed to maintain a certain level of consistency and to support synchronization. Existing mobile computing middleware systems do not address this issue and a consortium (i.e, SyncML) has been established in order to provide standards for synchronizing data in mobile computing. SynchML provides a set of specifications for the standardization of synchronization of data (in any format) between different devices, using WSP, HTTP, or Bluetooth protocols. XMIDDLE uses tree structures for representing data and defines protocols that take advantage of this format. SyncML focuses on peer-to-peer synchronization, where a client/server relationship is always established among the devices. No ad-hoc networking setting is supported by SyncML, whereas XMIDDLE also supports reconciliation of different clients that possess replicas of specific branches of an XML tree. SynchML also defines reconciliation policies for data synchronization. However, the polices are either on the server or client side. The case in which the client wants to indicate how to reconcile data to the server is not supported. Hosts sometimes need to specify different reconciliation policies and some priority structure among the policies is needed to actually choose which policy to apply. Unlike SyncML, XMIDDLE avoids the need for application to log every change they apply to shared data. Instead XMIDDLE uses a versioning system to make this aspect transparent [11]. SyncML, on the contrary, leaves the logging to the application level. Security and authentication aspects are investigated in the SyncML specification which XMIDDLE does not tackle yet. However some authentication mechanisms similar to the one of SyncML could we be put in place in XMIDDLE, too.

The XMIDDLE strategy for data synchronization exploits well established techniques and tools for replication and reconciliation on trees. In [16] some formal work on application-independent reconciliation has been carried out, which also focuses on a structured way for applications to influence data reconciliation choices. XMIDDLE exploits semantic knowledge about element types; a set of reconciliation primitives is defined in XMIDDLE and the mobile application engineer can specify the way these primitives are combined to determine an application-specific reconciliation policy. In this way we can ease the burden of applications, relying as much as possible on the middleware, while, at the same time, providing for the application semantics and user policies. This differentiates XMIDDLE from systems like CVS and Bayou. CVS is a source code versioning tool that leaves everything in the hands of the user; conflicts are detected based on updates done in the same line of the file by different users, and the conflict resolution is left to the user. Bayou reconciles application-specific information in an application-independent way, preventing the application from influencing the outcome of the reconciliation process. Bayou's philosophy is the traditional middleware one, which calls for complete *transparency*.

6 Conclusions

The growth of the recent mobile computing devices and networking strategies call for the investigation of new middleware that deal with mobile computing properties such as disconnection, low/expensive bandwidth, scarce resources and in particular battery power, in a natural way. XMIDDLE is one possible answer to these needs that focuses on data replication and synchronization problems and solves them exploiting reconciliation strategies and technologies.

The implementation of the current prototype of XMIDDLE [17] is based on Wireless LAN and UDP, however we plan to migrate the system to Bluetooth for more testing.

XMIDDLE is an example of a reflective middleware [5]. XMIDDLE abandons replication transparency as we believe that in the challenging mobile computing environments middleware systems have to take advantage of application-specific information to achieve an acceptable performance, usability and scalability. We consider our effort on XMIDDLE to be just the first step in that direction and believe that a number of other forms of transparency have to be given up, too. Location transparency, for example may have to be discontinued to provide location aware services. In general, this will lead to a new class of context-aware applications [3, 9], which can influence the way middleware implements interactions between mobile components based on the context in which the components operate.

Acknowledgements: We would like to thank Stefanos Zachariadis for his participation in the implementation of XMIDDLE and for his suggestions.

References

[1] V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood. Document Object Model (DOM) Level 1 Specification. W3C Recommendation http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001, World Wide Web Consortium, Oct. 1998.

[2] G. Cabri, L. Leonardi, and F. Zambonelli. Reactive Tuple Spaces for Mobile Agent Coordination. In *Proceedings of the 2nd International Workshop on Mobile Agents (MA 98)*, number 1477 in LNCS. Springer, 1998.

[3] L. Capra, W. Emmerich, and C. Mascolo. Middleware for Mobile Computing: Awareness vs. Transparency (position paper). In *Int. 8th Workshop on Hot Topics in Operating Systems*, May 2001.

[4] J. Clark and S. DeRose. XML Path Language (XPath). Technical Report http://www.w3.org/TR/xpath, World Wide Web Consortium, Nov. 1999.

[5] F. Eliassen, A. Andersen, G. S. Blair, F. Costa, G. Coulson, V. Goebel, O. Hansen, T. Kristensen, T. Plagemann, H. O. Rafaelsen, K. B. Saikoski, and W. Yu. Next Generation Middleware: Requirements, Architecture and Prototypes. In *Proceedings of the 7^{th} IEEE Workshop on Future Trends in Distributed Computing Systems*, pages 60–65. IEEE Computer Society Press, Dec. 1999.

[6] E. Freeman, S. Hupfer, and K. Arnold. *JavaSpaces[tm] Principles, Patterns, and Practice*. Addison-Wesley, 1999.

[7] IBM. T spaces. http://almaden.ibm.com/cs/TSpaces.

[8] T. Imielinski and B. R. Badrinath. Mobile wireless computing: challenges in data management. *Communications of the ACM*, 37(10):18–28, Oct. 1994.

[9] L. Capra and W. Emmerich and C. Mascolo. Reflective Middleware Solutions for Context-Aware Application . In *3th International Conference on Metalevel Architectures and Separation of Crosscutting Concerns (Reflection 01)*, LNCS, September 2001. To Appear.

[10] E. Maler and S. DeRose. XML Linking Language (XLink). Technical Report http://www.w3.org/TR/1998/WD-xlink-19980303, World Wide Web Consortium, Mar. 1998.

[11] C. Mascolo, L. Capra, and W. Emmerich. XMIDDLE: A Middleware for Mobile Ad-Hoc Networking . Technical report, University College London, Dept. of Computer Science, 2001. Submitted for Publication.

[12] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[13] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Flexible Update Propagation for Weakly Consistent Replication. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16)*, pages 288–301. ACM Press, 1997.

[14] G. Picco, A. Murphy, and G.-C. Roman. LIME: Linda meets Mobility. In *Proc. 21st Int. Conf. on Software Engineering (ICSE-99)*, pages 368–377. ACM Press, May 1999.

[15] M. Satyanarayanan. Mobile Information Access. *IEEE Personal Communications*, 3(1), Feb. 1996.

[16] M. Shapiro, A. Rowstron, and A. Kermarrec. Application-independent Reconciliation for Nomadic Applications. In *Proceedings of European Workshop:"Beyond the PC: New Challenges for the Operationg System"*, Kolding, Denmark, 2000. SIGOPS.

[17] S. Zachariadis. Implementing XMIDDLE, an XML-bsed Platform for Mobile Computing and Ad-Hoc Networking. Technical report, University College London, Dept. of Computer Science, 2001. Final year project.