A Pervasive Synchronization Middleware for Mobile Devices

Puneet Gupta, Zalak Dedhia, Atanu Roy Chowdhury, Kartik Muralidharan
Convergence Research Group,
Software Engineering and Technology Labs
Infosys Technologies Ltd, Bangalore 560 100, India
{Puneet_Gupta,Zalak_Dedhia, Atanu_Chowdhury & Kartik_Muralidharan}@infosys.com

***Abstract*— The evolution of wireless communication, and the protocols built around them, has empowered a significant proportion of computer users to go cable-less. However there seems to be ample opportunity to improve upon its promises of flexibility and seamlessness. Moreover as computing devices become abundant, they intrude into several facets of our life, bringing with it a deluge of possibly redundant information. Being aware of the decreasing probability of the end user being a computer wizard, it becomes necessary to isolate system complexity from the end user. So here in this work we discuss our architecture for a synchronization solution, to keep the data on a mobile device fresh. A context aware middleware, plays a crucial component of the architecture.**

I. Introduction

A simple walk through an electronic gadget shop today is sure to overwhelm you with a plethora of acronyms on wireless access technologies. A non-exhaustive list would possibly include Wi-Fi, Wi-Max, BlueTooth, FireWire, GPRS, ZigBee, and the list continues to grow. However most of these technologies are not interoperable. This invariably leads to a partitioning of wireless devices into various 'worlds' which are accessible by only a particular technology. Thus a group of devices in a BlueTooth scatternet will remain oblivious of a spatially collocated WLAN environment!

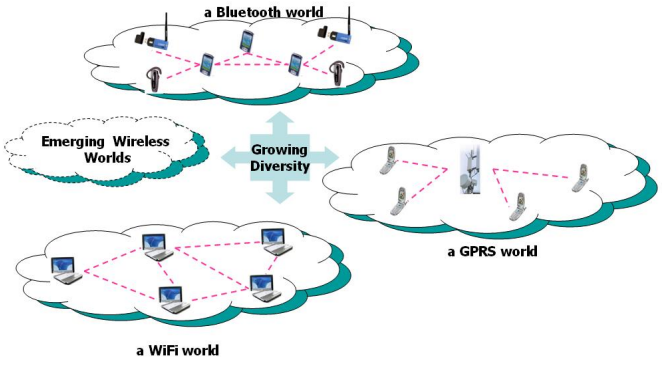

Fig. 1. Diversity in the Wireless World

The two worlds cannot intercommunicate unless there exists a translator device (or gateway) with the physical capabilities to access multiple worlds. Interestingly most hardware manufactures have begun to incorporate this necessary implication of the above mentioned scenario, in their latest products like Palm Treo 650[1], HP iPaq h4350 [2] and iPaq 4155 [3]. Although we are progressively transposing towards a convergent ambience, as of date mobile devices continue to exist in multiple 'worlds'. Equally disparate are their form factors, computational capabilities, memory constraints, battery backup and even their displays.

Moreover, in the context of seamless computation, there is also a pressing need to interact between multiple data source types, distributed across various domains. Thus we observe that advanced wireless access techniques empower us with greater mobility and flexibility, but also curtails the performance of mobile devices. Some of these limitations are highlighted in Table I.

In today's context, mobile computing is more about extended cable-less operations rather than being truly pervasive. Pervasive computing aims provide an user with a completely transparent environment where he cannot even perceive the existence of non-trivial computing in his surroundings, let alone imagine its complexities.

Since this work was done in context of an enterprise IT solution, the next section showcases a roadmap for enterprises to acquire pervasive capabilities. The penultimate section delves on the evolution and architecture of our pervasive synchronization middleware.

II. Pervasive IT

The concept of pervasive computing is a paradigm

[1]http://www.palm.com/us/products/smartphones/treo650/
[2]http://www.bargainpda.com/default.asp?newsID=1620&showComments=true
[3]http://www.mobiletechreview.com/ipaq4150.htm

TABLE I
Limitations in mobile devices

Wireless Medium	Packet Loss Due To Transmission Errors Frequent Disconnections/Partitions Limited Communication Bandwidth Broadcast Nature Of The Communications Security Concerns Half Duplex Transmissions
Mobility	Dynamically Changing Topologies/Routes Lack Of Mobility Awareness By System/Applications
Lightweight Terminals	Short Battery Lifetime Transmission Ranges Processing Capacity

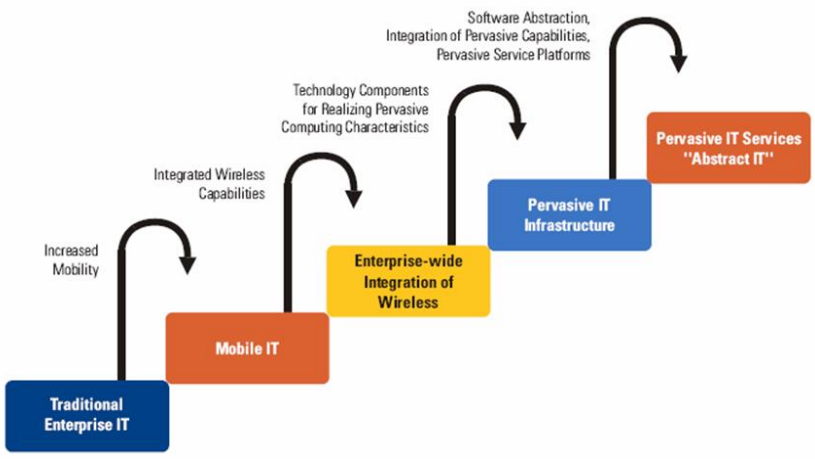

Fig. 2. Roadmap for attaining Pervasiveness in IT

shift [1] [2] from a traditional IT point of view, including to a large extent mobile IT. However, from an enterprise point of view, treating pervasive IT as a radical departure from the traditional IT presents a dilemma. While on the one hand, there is an opportunity to derive productivity and business benefits from an intelligent IT infrastructure built on the pervasive computing paradigm, on the other hand, there is a need to protect investments already made in the existing IT infrastructure, including mobile IT. It is in this context that we suggest the enterprise pervasive IT model (EPIM) [3] as a gradual evolution from an existing IT infrastructure point of view. This evolution is captured in Fig 2[4].

A. The Mobile User:Today and Tomorrow

Recent research findings from leadings analysts like Forrester and Gartner point out that market leaders across almost every business vertical are migrating to some form of mobile solution. We also observe that the migratory patterns can be classified as client facing mobile solutions (eg SMS notification of a banking transactions) or solutions to improve internal agility (eg connectivity with field force).

Whereas the first group of solutions allow an informed end user to choose a mobile device, the second category proactively places complicated technology with inept hands. If this trend continues, and we have no reason to believe otherwise, in future only an insignificant proportion of users can actually comprehend the computational complexity of the processes in their device. In fact this transparency must necessarily remain so as to ensure the mass acceptance of the devices as well as the processes in them.

B. Need for a Pervasive Synchronization Middleware

Synchronization is backbone of the mobile applications due to simultaneous existence of data at multiple ends. It has made a dream, of data availability anytime anyplace, come true. But it doesn't come without consequences such as complexity, redundancy, failures, security, and validation, supporting disparate technologies, data stores, and devices and so on. To avoid getting mired with these varied complicated synchronization facets a pervasive synchronization middleware seems must, that will seamlessly abstract difficulties and expose simplicities. Not just this but there can be many reasons that would drive us to the need for the existence of a middleware (Fig 3).

Let us ponder over the necessity of having it before delving into the intricacies of the proposed architecture.

- Isolating both the end user and the developers from the complexities of the underlying systems.
- Secondly, a resource constrained mobile device can only provide a limited or local view which quickly becomes stale. To ascertain freshness, updated information must be pulled from the neighborhood at periodic intervals or on demand.
- In other scenarios, the device might not be able to show the entire database at once. Therefore it becomes mandatory to not only tag available information but also know as to where the required information can be found.
- Finally, absence of centralized information retrieval has the inherent advantage of no single point of failure and high scalability.

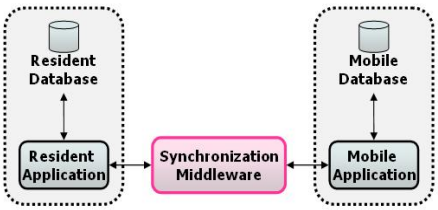

Fig. 3. Overview of Pervasive Synchronization Middleware

[4]The EPIM model forms the basis of convergence research at Infosys. Our Synchronization middleware is a result of implementing technology components, which in turn realize pervasive computing characteristics in the enterprise (fourth) block of EPIM

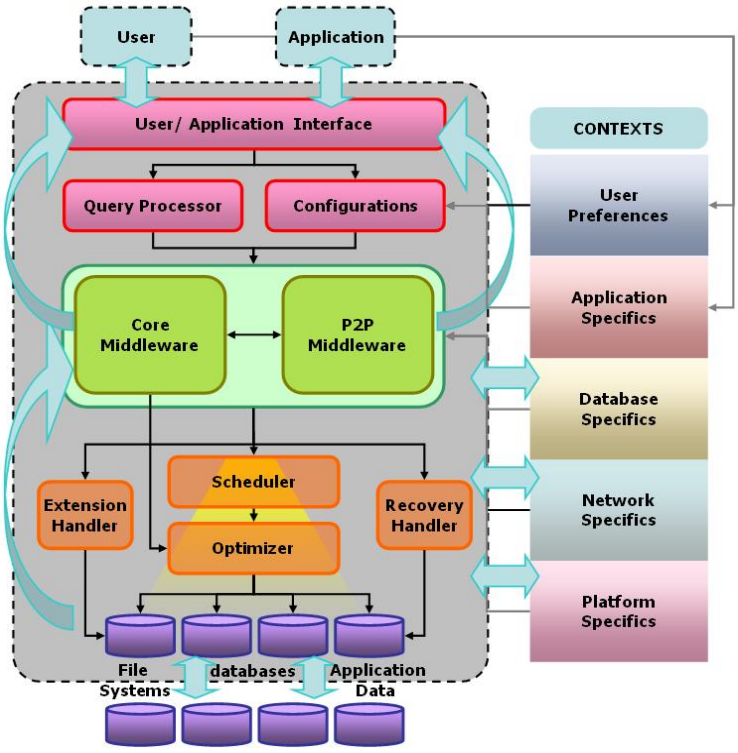

Fig. 4. Architecture of Pervasive Synchronization Middleware

III. Evolving The Pervasive Synchronization Middleware

Most of the block names capture the essence of their purpose. For example, the Interface module exposes relevant APIs to the developer, whose application overlays interact with the end user. Similarly the Query Processor is able to parse high level query constructs, whereas the configuration module customizes the middleware for individual user preferences. Other blocks require elaborate explanations and are therefore included in the following subsections. Data and control information pass through these blocks as shown in Fig 4.

Fig 5 captures the application interfaces as exposed to the developer. All that is required to synchronize two disparate data sources are the source and destination database type and location as well as the table name. Security credentials if applicable can also be specified. The middleware abstracts the diversity of the database types and makes the information available locally. We show synchronization between a Microsoft Access database type at the server end and an SQL CE database residing on the mobile device. The last panel shows the information retrieved from the local storage after synchronization.

A. Context Sensitivity

Pervasive computing is all about harnessing the computational power of distributed computing devices. Therefore, any pervasive computing artifact must essentially adopt a context sensitive approach rather than exhaustive searches. 'Context sensitivity' can be defined as the change in the response of a deterministic device, in accordance with the parameters which qualify the situation. It is to be noted that 'context sensitivity' is a property of the system and not a task.

Our architecture is characterized by its ability to respond to a given set of 'contexts'. Figure 6 lists these contexts and factors which affect them. For example, an important network context is the occurrence of "Occasional Connectivity" wherein the user experiences short term disruptions in the connection. Under normal circumstances the transaction submitted just prior to (or during) this disruption is most likely to get dropped. Moreover the user remains unaware of this drop, as the disruption periods are extremely short. The proposed middleware takes care of such situations by incorporating intelligent store and forward mechanisms that ensures proper execution of the submitted transaction. Prioritization of submission can be achieved through transaction sensitivity associations. Note that all this is done without the user intervention.

B. The Core middleware

The functionality of the core is controlled by the context plane. Some contexts, like assessing rights of a particular user, indirectly affect the Core by modifying the Configuration module. In other cases, contexts like available NICs have direct implications in the downstream control of the Core.

The functionalities of the Core can be listed as follows:

- Specifying the appropriate databases, along with the parsed query, to the query Scheduler.

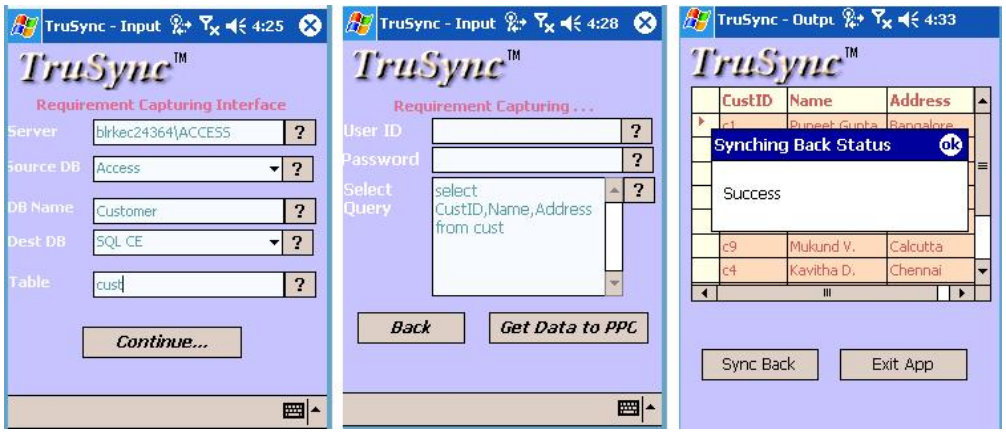

Fig. 5. Developer Interfaces

User Preferences	• User Profile • User Authorization (passwords/ passcodes) • Degree of Detail (header/partial/ complete) • Privacy Constraints (secure / normal)
Application Specifics	• Sensitivity of Data (real time constraints/ jitter constraints) • Periodicity of Traffic (bursty, CBR) • Priority of Application
Database Specifics	• Data retrieval (block size/ time constraints/ other nuances) • Commit Status (ability to track changes) • Transaction Status (check pointing/ full or partial rollback) • Algorithmic complexities (search/ sort/ insert/ update/ delete)
Network Specifics	• Throughput (constant/ choppy) • Connection Status (always connected/ occasionally connected) • Link Reliability (fast unreliable channels used for short transfers) • Peer Status (standalone/peer group/ 1-hop wired connectivity available)
Platform Specifics	• Device Type (server/ desktop/ laptop/ PDA/ mobile) • Available Power (always on/mostly on/occasionally on) • Storage Capacity (tape drive/ hard drive/ primary memory/ flash drive) • MAC Technology (Ethernet/ WiFi/ WiMax/ Bluetooth/ GPRS)

Fig. 6. Context Parameters

- Specifying communicating interfaces and other relevant information to the query Optimizer.
- Issuing synchronization commands for databases with stale data.
- Specifying the master repository for synchronizing a local database. In case the master repository is more than a hop away, it passes control to the P2P middleware.
- Control the Recovery Handler to resolve conflicts at the earliest.
- Control the Extension Handler to accommodate new databases and minimize complexity for the developer.
- Retrieve Information from the local databases and pass them onto the user interface.

C. The P2P middleware

The P2P middleware comes into effect when communication needs to be established between devices that are more than one hop away. There are two distinct aspects to this problem. To optimize communication overheads we need to determine which peer has access to the main repository. In keeping with the current trends of P2P networks, we feel a dynamic hash table (DHT) based approach is most suited. The other aspect is the communication itself, because we are unsure of the accessibility of the lowest layers. In this context we observe that a web service (WS) based solution can overcome these nuances.

In the WS approach, servers are configured to behave as webservers whereas resource constrained devices have lightweight webservers, serving as proxy to the main server. Thus the communication between two heterogeneous devices is through a set of XML based protocol, where a reliable asynchronous messaging system like SOAP is used to connect to webservices hosted by lightweight or full-service webservers. This architecture is shown in Fig 7.

D. Observations

Since perpetual connectivity cannot be guaranteed, therefore databases are to be synchronized opportunistically, whenever the occasional connection is available. Secondly, the entire endeavor is to make the application process believe that all data is being retrieved from a local database. However a node cannot deny responsibility of maintaining the DHT of its peer group. Thirdly, network topology forms a context factor and in that effect alters the behavior of the middleware itself.

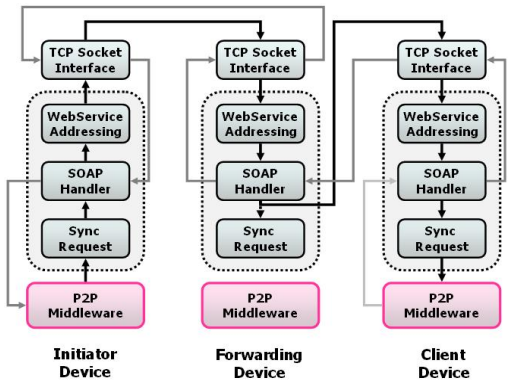

Fig. 7. Architecture of Web Service based Communication Module

Fourthly, we must take note of the fact that a mobile device is constrained in its memory and therefore necessitates explicit buffer management. To do this it is must that we be able to effectively filter inconsequent messages , with respect to the current user.

E. Application Scenarios

The WS architecture immediately allows different wireless technologies to have a more meaningful and symbiotic coexistence. Therefore the scenario depicted in Fig 1 transforms into the one depicted in Fig 8.

Consider a field force deployed at a faraway location from the corporate data centre, where the proposition of providing every individual with GPRS connectivity is an expensive one. However the team leader can be provided with a connection, and all synchronization now happen when the team gets together over coffee. This scenario is depicted in Fig 9.

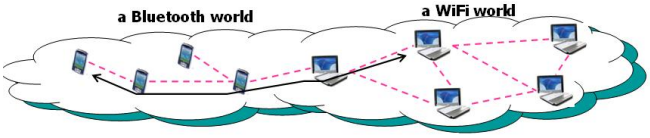

Fig. 8. Combining Worlds

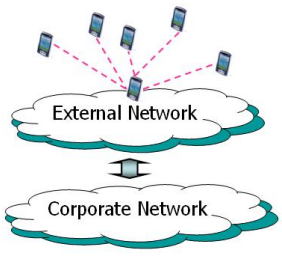

Fig. 9. A field force Use Case

IV. Conclusion

In this paper we have presented the architecture of a pervasive computing artifact that effectively uses context information to enhance user experience. We have also presented the architecture of a web service based communication model that allows intercommunication between disparate wireless worlds.

References

[1] M. Weiser, The Computer for the Twenty-First Century, Scientific American, pp. 94-10, September 1991.

[2] M. Satyanarayanan, Pervasive Computing: Vision and Challenges, IEEE Personal Communications , August 2001.

[3] Puneet Gupta, Deependra Moitra, "Evolving a pervasive IT infrastructure: A technology Integration Approach", ACM personal and Ubiquitous Computing, Vol 8(1), Feb 2004, pp 31-41.

[4] C. Bisdikian, I. Boamah, P. Castro, A. Misra, J. Rubas, N. Villoutreix, D. Yeh, "Intelligent pervasive middleware for context-based and localized telematics services," Proceedings of the second international workshop on Mobile commerce, ACM Press, pp 15-24, 2002.

[5] L. B. Mummert, M. R. Ebling and M. Satyanarayanan, "Exploiting weak connectivity for mobile file access. Proceedings of the fifteenth ACM symposium on operating systems principles," December 3-6, 1995, Copper Mountain, Colorado. Pages 143-155.