Mobile Networks and Applications 9, 651–662, 2004

Disconnected Processes, Mechanisms and Architecture for Mobile E-Business

J. SAIRAMESH, S. GOH, I. STANOI, S. PADMANABHAN and C.S. LI
Institute for Advanced Commerce, IBM T.J. Watson Research Center, Hawthorne, NY 10532, USA

Abstract. With the tremendous advances in hand-held computing and communication capabilities, rapid proliferation of mobile devices, and decreasing device costs, we are seeing a growth in mobile e-business in various consumer and business markets. In this paper, we present a novel architecture and framework for end-to-end mobile e-business applications (e.g., point of sales). The architecture takes into consideration disconnection, application context, synchronization, transactions and failure recovery modes to provide mobile users with seamless and transparent access to business transactions and business-context specific data. In our architecture, we consider a novel business process design based on state-machines and event management to handle disconnection, resource limitations and failures. We designed, implemented and deployed a system for mobile e-business on clients (e.g., PDAs and PocketPCs) integrated with private exchanges and sell-side servers. The state-machine model with failure recovery mechanisms enables handling of one-to-many and many-to-one disconnections in large mobile e-business environments. The e-business framework on mobile clients is implemented based on J2ME, Webservices, and open XML standards. A detailed performance study of commerce transactions was done on different mobile client devices with diverse computing, memory and storage capabilities. We compare the performance of a purchasing application and the middleware on various devices such as PDAs and Laptops. We demonstrated that for small devices with limited capability the performance is reasonable. For devices with more computing capability, the response time is excellent.

Keywords: mobile e-business, mobile disconnection, remote disconnection, failure recovery, seamless business transaction, mobile commerce

1. Introduction

With advances in computing and communication capabilities of hand-held devices such as PDAs, Pocket PCs, wireless enabled Laptops and others, we are beginning to see an emergence of a variety of applications in mobile e-business (m-business). These mobile devices are becoming as powerful as laptops and desktop PCs, thereby attracting consumers and businesses to use them in their daily business and recreational activities. We consider that mobile e-business is a major business opportunity in the coming years due to the flexibility and cost-performance trade-offs. Currently various groups and companies are experimenting with mobile applications in order to validate the possibility of using these devices for all kinds of core business activities within an Enterprise and across enterprises (in the supply and value chains).

The market for mobile phones and hand-held devices is exploding given the low costs, increasing computational power and ease of use. The current estimate of number of devices sold is around 500 million [3], and growing to a billion in the next couple of years. In contrast, the number of desktop PCs has saturated to around 500 million [3,12]. Most mobile users will at some point in the future have more than one mobile device and will increasingly depend on these devices for accessing the Internet and performing daily social and commerce activities. For businesses, the low costs and increased capacity provides a tremendous opportunity to exploit these mobile devices for routine B2B [3,12,18] activities such as purchasing, point of sales, technical services, inventory tracking, order status and order notifications, warehouse management and others.

With the emergence and rapid early deployment of Wi-Fi networks and Blue-Tooth LANs, a number of new and ingenious possibilities of connecting wireless devices such as PDAs, PocketPCs, wireless Laptops to the local enterprise networks and the Internet have emerged. Businesses are realizing the potential of such connectivity and capability to enable a range of activities for their mobile employees. In addition, many businesses (including cafes, airports) are beginning to capitalize and deploy wireless networks to the needy business traveler. In the following we present some of the motivational factors for our research work to enable e-business applications for business users.

1.1. Motivational factors

Reduction in costs. The key motivation in our view is one of economics, playing a role in bringing low cost computing to the consumer whether a business or an individual, who can easily access the internet through a mobile device. Internet enabled mobile devices are gaining popularity as the costs of subscription are getting lower, thus making a case for businesses to invest in mobile applications where needed [2,3,13].

Improved computing capabilities. Another motivating factor has been the increasing computing power and storage in these hand-held devices, making them almost as powerful as laptops and desktops. Most of the hand-held devices (such as PALM and PocketPC devices) offer sufficient of (16–64 Mbytes) memory, large enough to support embedded operating systems, small footprint databases, web-browsers, and other applications. Some of the recent devices have computing capabilities between 200–400 MHz of processing capacity, very close to where laptops were a few years back. In addition, laptops are being wireless enabled in business environments, providing tremendous capabilities for client side mobile e-business.

Ease of access to content and transactions. With the increasing capabilities of the devices, many businesses are experimenting with supporting content and transaction based applications, including email and others on mobile devices in order to provide flexibility sales agents, technical service representatives, retail point of sales agents and others. This enables the mobile workforce to perform their business activities efficiently without having to be tied to an office or desk for accessing enterprise intranets. The "challenges" are many for accessing content and performing transactions. In a large Enterprise, many back-end systems need to be integrated for business processes to run efficiently. For the same business process to run on a mobile device, the device has to connect to multiple back-end systems and transact with each one of them. We term these interactions as one-to-many connections, and the challenge is to consider disconnection with one or more of these back-end systems.

Standards for programmability. There is a push by many technology companies to deploy a programming base for applications on mobile devices. The application programming frameworks in devices that are gaining popularity are the following: J2ME based programming environments on Linux, WinCE and Palm OS environments. In addition, there is tremendous push towards standards of programming and APIs on the client devices to enable interoperability of applications. Interoperability is a key challenge for mobile e-business on clients, and they have to interact with multiple back-end business critical applications to run the business process.

Demand for applications. With the increasing capacity and low costs, the devices are ideally suited for business-to-business applications such as purchasing, transportation capacity tracking, inventory management, point of sales, promotions, order status information, notifications and others. Businesses that have employees who are mobile most of time as in the following industries: agriculture, transportation and logistics, warehouse management and other are increasingly demanding mobile devices that provide very specific business function.

1.2. Contributions

In this paper we present our mechanisms and architecture for enabling mobile e-business applications under disconnection, resource failures and resource constraints. Our contributions are as follows:

- A novel business process design based on state-machine models, which takes into consideration disconnection modes, synchronization, failure events and resource constraints. With this design, business processes on a mobile client are run even under system and network abnormalities and failures. This provides the end-user the flexibility to continue doing the commerce and business transactions while being disconnected and still maintain the application context when connected to the network.
- Why is this novel and new? Pioneering work was done in Coda [14,15] for file-systems under disconnections and unreliable networks. Our work is different in many ways, it is focused on handling disconnection of business processes based on the application context which is dependent on application data, state of the user application, and business logic associated with the business process. Firstly, the work done in Coda [14,15] has been in handling disconnection for file systems where application specific business processes and semantics of the execution of the business process are not considered. Secondly, the state-machine model we propose in this paper handle "one-to-many" disconnections, where one mobile client can interact and handle disconnections with one or more of them. This is in our view is critical for mobile e-business applications where multiple back-end systems are integrated for enterprise business processes. Thirdly, we consider state-machines for every business process (fine-grained model) running on the mobile device, and when disconnection occurs we maintain the respective states of the business process so that we can recover from that application state. These three reasons in our view differentiate our work from Coda file-system.
- A programming environment based on J2ME for e-business applications. The programming environment provides framework for developing and deploying applications easily. The framework provides the following: persistence for business objects, business logic commands, event management for handling all kinds of events (including failure events), a process engine, and a messaging system to enable integration with external severs. The programming model extended the traditional models by taking into consideration disconnection and synchronization in every business process execution.
- An architecture for handling disconnected computing, synchronization and transactions with business processes on the client-side. The e-business components include the following: catalogs, membership, shopping cart, order management, promotions and contracts are provided as a part of the framework to build e-business applications. The underlying middleware on the client handles disconnection, synchronization and failure recovery independent of the application context. The scheduling mechanisms enable the mobile device to automatically launch incomplete business processes just after the disconnection is cleared, and connection is active. This provides a certain level of self-managing capability which is unique.

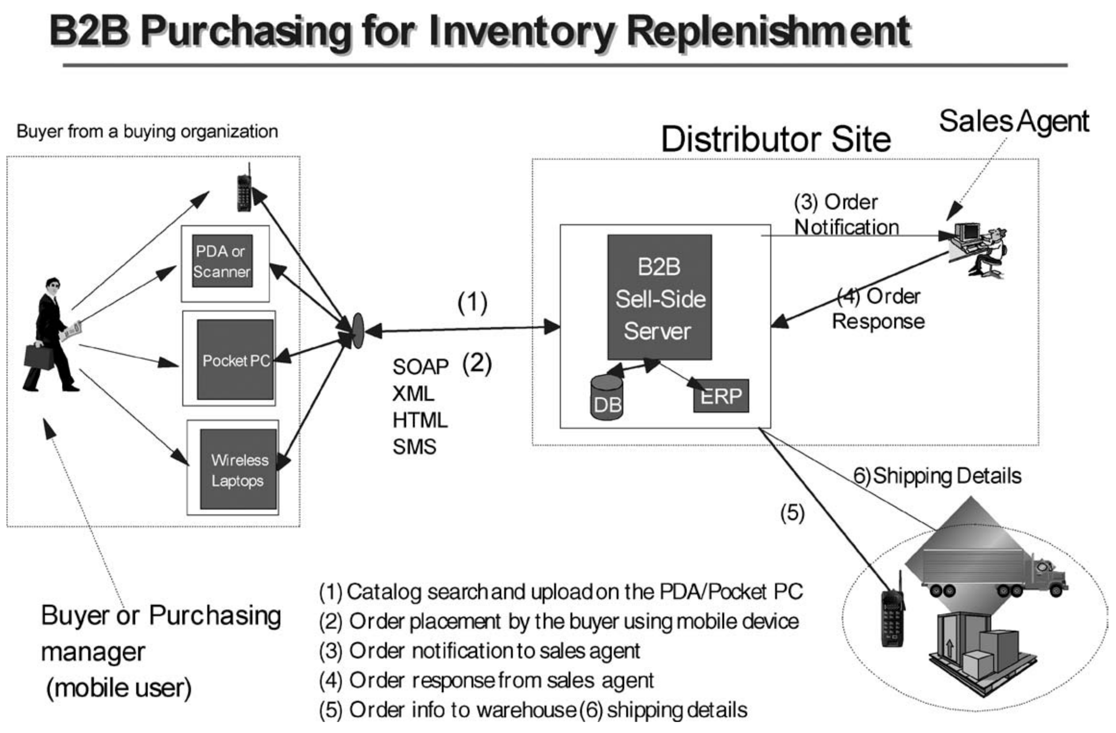

B2B Purchasing for Inventory Replenishment
Buyer from a buying organization
PDA or Scanner
Pocket PC
Wireless Laptops
Buyer or Purchasing manager (mobile user)
(1)
SOAP (2)
XML
HTML
SMS
Distributor Site
Sales Agent
(3) Order Notification
(4) Order Response
B2B Sell-Side Server
DB
ERP
6) Shipping Details
(5)
(1) Catalog search and upload on the PDA/Pocket PC
(2) Order placement by the buyer using mobile device
(3) Order notification to sales agent
(4) Order response from sales agent
(5) Order info to warehouse (6) shipping details

Figure 1. Purchasing scenario.

- We propose a caching and synchronization model for client data that is refreshed based on expected expiry and costs of retrieval. Why is this new? Though substantial work has been done in caching and synchronization for web-based pages and objects in distributed systems, not much work has been done to cache objects based on the "semantics" of a business process and the application context. The caching and synchronization is done in the context of disconnection in order to keep the business process transparent. The novelty of the caching model is based upon state-machine approach, where the business process state, business data are cached for providing transparency during discussion. The mechanism design considers timestamps and freshness of the data to constantly update the client data whenever the connections to the server are established.

In section 2, we present some business scenarios.[1] In this section we also present the design issues for client side data and transactions. In section 3, we present the model of a connected and disconnected process, and we present our mechanisms for handling disconnections, failures, and recovery. In section 4, we discuss the various design criteria for middleware architecture on a variety of mobile clients. The architecture includes a process engine that drives the business processes on the client side. In section 5, we present a configuration for end-to-end mobile e-business via wireless (Wi-Fi) networks and we study the performance of this configuration for two e-business applications over two different wireless devices.

[1] The scenarios are based on some customer requirements for their sales and purchasing departments.

2. Business scenarios: purchasing

In this section, a business scenario is presented for mobile e-business applications in retail and wholesale distribution. In figure 1, a purchasing scenario is illustrated.

2.1. Scenario under connectivity

Consider a purchasing manager at a car dealership, checking inventory of tires. The manager finds that the inventory on tires from manufacturer X is low, and wishes to place an order to a distributor of those tires. Assume that the manager has registered the dealership at the distributor website. The manager connects through a personal PDA or PocketPC from the warehouse and retrieves part of the distributor catalog on tires for price and availability information. The PDA has a local catalog (or a local cache) that keeps a copy of retrieved distributor catalog. The purchasing manager browses the local catalog, adds items to a shopping cart on the device database, and places an order.

If the device is connected, the order is sent away to the distributor site server that processes orders placed by purchasing agents. Once the orders are sent (reliably) the mobile device waits for a confirmation that that the order will be fulfilled partially or completely, and other information such as delivery dates and location of shipment. The distributor organization could have a sales agent who manually browses through some orders before accepting them (based on the some rules). The order response and final confirmation is sent back to the buyer's device via notifications. The order information is then sent to a delivery or transportation company for final delivery to the buyer.

2.2. Disconnection and failure scenarios with context

The above scenario assumes that that there are no resource failures on the PDA, and network disconnection and network failures. We now consider the same scenario with failures that could occur during any stage of the purchasing process.

> Consider the purchasing scenario as described above. The purchasing manager views the local catalog that was downloaded from the distributor website. The manager places an order with multiple items to a distributor who accepts orders on-line. Many failure events could occur between the time that a part of the catalog was downloaded from the distributor website and the response from the distributor website for the order placed by the purchasing manager. Consider that the device was in connected mode when the catalog on tire prices was being retrieved. Consider now that during order submit phase of the business process, disconnection occurs.
>
> If the device is connected, the order is sent right away to the remote distributor website for order fulfillment. The mobile device then waits for an order response acknowledgement from the distributor web site. The order confirmation may not come back immediately from the seller because the sales agent on the sell-side could spend some time processing the order and manually accept, partially accept or reject the order. If there is network disconnection, the responses to the order from the distributor web site can arrive asynchronously depending on when the connectivity comes back fully. The device has to maintain context information on which orders submitted and match the responses that come back from the distributor.

2.2.1. Issues on client side for transactions

Using the above scenario, the following are some of the core issues to consider when building content and transaction based applications (e.g., purchasing) on the mobile client device.

Disconnection could occur during any stage of the purchasing process due to many reasons such as: network timeout or network failure, low battery, local memory constraints and other system failures. For example, the mobile device could be disconnected from the network when retrieving catalog entries from the server, or the device could be disconnected when trying to send a completed order form to the distributor server. In addition, the communication network could be slow enough to cause a time-out at the client side, causing disconnection. Another reason for disconnection can be the cost of access during peak hours (if using a modem attached to the PDA). It is beneficial to be disconnected till the network bandwidth and/or cost improve. Most network providers advertise the cost of data transfer (data per byte) for various time-periods during the day and month. This information could be used to retrieve non-critical catalog data in lower cost time-periods of the day.

The order confirmation for various orders and order-items within an order could arrive as one message or could arrive as a sequence of messages asynchronously from the sell-side server.[2] The application on the mobile device has to maintain local order context when an order confirmation has been received from the server. The client device has to maintain session information with the server to have speedy and smooth order transaction. For this the device has to maintain cookie information and server specific information on the devices such as the authentication information of the mobile user when connecting to the distributor web site.

Related work spans over multiple research fields from workflows [1,4,8], transaction management [2,11], recovery [10], and caching [9,17]. A new standards body on synchronization is emerging from corporations who want interoperability in synchronization of data [3,13,16]. In this paper, we focus on the problem and issues around transactions on the clients such as creating and placing an order on the client and submitting that order to a commerce server for fulfillment.

2.2.2. Issues of managing client data

The design of a mobile framework for e-business has to also take into account the migration and storage of business processes (in XML format) and application context data from server to the client. From the perspective of the user, information on the mobile device should be available, up-to-date, and consistent with the server data. Obviously, there are challenges in the data management of a mobile device that have not been addressed in more traditional systems. The following is an overview of some of these challenges that we further address in more detail:

- *Transparency*. Unless explicitly requested, provide the user with as little knowledge about access to data and communication between client and server. There are limitations to what can and should be supported by the framework. The system should:
  - ∗ Allow users to explicitly request access to fresh, server data.
  - ∗ Open communication with the server, at minimal cost to the user. This is an important aspect, because the user may pay for the transfer of information with the remote server.
  - ∗ Take into consideration that any background process may interfere with processes explicitly initiated by the user. On one hand, background processed can obviously use resources that the user needs. On another hand, since these processed affect the same data that the user may access, there are concurrency control problems that have to be resolved.
- *Efficiency*. It is important that data cached on the client, be as up to date as possible. An example is the high cost of aborting distributed transactions, which can be avoided or reduced by having access to updated data on the client.
- *Synchronization*. Synchronization refers not only to updating local data, but also resolve update conflicts between the local and remote versions of the same object.

[2] The sales agent at the distributor e-business site might have to verify the inventory levels before accepting or rejecting an order from the buyer.

- *Advanced transaction models*. There is a conflict between trying to reduce the number and size of messages between client and server on one hand, and reduce the abort rate of distributed transactions on the other hand.

2.2.3. Issues on server side
The server on the distributor web site handles external requests from all kinds of clients (web-browsers, PDAs, Pocket-PCs, wireless Laptops, other servers) as long as the user submitting the request is registered and is authentic. The server has to maintain profile information of the type of device that a user will use for placing orders. Based upon the type of device, the server could send the appropriate order confirmation message in XML or other user-interface format (e.g., HTML) depending on the preferences of the user and the capability of the device. The server has to authenticate and manage the session of disconnected clients, which could have established one or more long-running sessions. Managing the client profiles and providing the right information in the format is one of the core issues on the server side.

3. Approach and business process model

In this section we present an approach based on state-machines to handling disconnection, failures and resource constraints.

3.1. Configuration and assumptions

We consider a simple configuration of mobile devices connected to the Internet and to a collection of pre-selected set of commerce servers. Each device has capabilities to support a local database systems (e.g., DB2e from IBM [6]), local file system, messaging system, run-time support for programming (e.g., Java based run-time) and administration applications. The following are the two main entities we consider in our design of the overall architecture.

Mobile device. The mobile devices in this architecture have capability to browse XML, HTML or WML and HDML forms/data. The devices are connected to local wireless LANS such as 802.11b or through mobile modems or via mobile providers who support wireless gateways. The assumptions are that each user has one or more devices. Each device supports an operating system (such as PALM or WinCE or Linux) to enable rapid deployment of applications over the local embedded database system, messaging system and run-time environments.

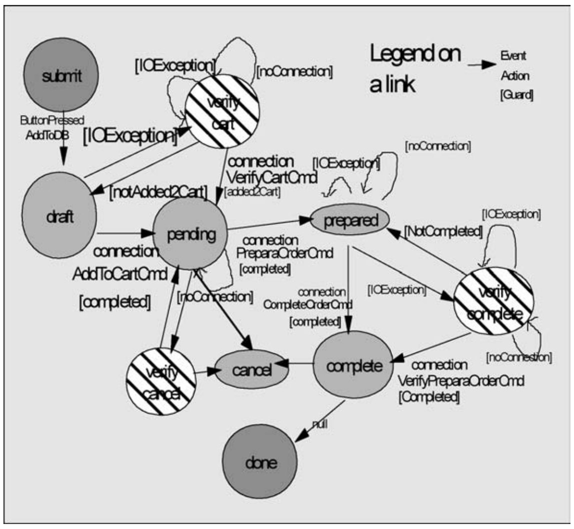
E-commerce and content servers. The e-commerce and content servers are web-application servers, which support J2EE based enterprise architecture. Requests sent to each server are handled by servlets [12,13], which parse the device context, and handle the incoming requests. Each server authenticates mobile users based on their profile and other information. We now proceed to define the process model for disconnection processes on the client side of the configuration.

3.2. Disconnected process state-machine

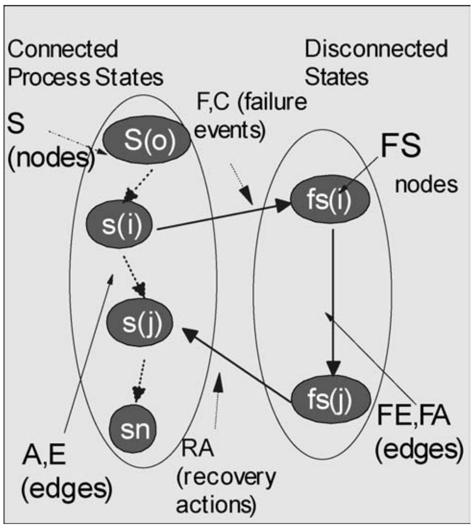
In this section we present our approach, which is a state-machine models of a business process. In the figure below, a process state-machine is shown which represents the business process on the client side (on the mobile device). A similar (if not the same) process description exists on the server side for the order process. The original process without disconnection consists of only 5 states (solid color) as shown in figure 2.

3.3. Client side state-machine

The client side the state-machine will be a subset of the process state-machine of the server. When an order is first created, it is in draft state (in the shopping cart), as shown in figure 2(a), and then it moves from state to state depending on the stage of the purchasing process. The following are the possible states the application process for order can go into



(a)



(b)

Figure 2. (a) order process state-machine and (b) general business process with disconnected states.

without considering the disconnection and other failure conditions: (1) Draft; (2) Pending; (3) Prepared; (4) Complete; (5) Cancelled.

The user of the mobile device could be processing one or more orders when disconnected. The orders submitted from the device could be in any of the five states. However, due to network disconnection, timeouts and failures, additional states represent the state of the order and the appropriate failure condition. The disconnected states are as follows: (1) VerifyCart; (2) VerifyComplete; (3) VerifyCancelled. When the network is disconnected the order process moves to one of these disconnected states. The state-machine which includes the actual business process states and the disconnected states while processing the order is shown in figures 2(a) and (b). The process engine (discussed in detail in section 4) drives the application from start to finish. From the time the user add items to the shopping cart to the order confirmation, the process engine executes the state-machine representation of the order management process on the mobile client.

The user of the application on the mobile device is transparent to whether the system is connected or disconnected. The underlying run-time ensures that the application process reaches the complete state once the connectivity is established when failures are repaired. Once the system conditions get back to normal, the process engine gets notified via an event. The event causes the process engine to move the business process from the "disconnected" state to the appropriate connected business process state by running the recovery procedures.

3.4. Server side state-machine

The server side processing of the state-machine is the following. The order processing states are as follows: (1) Draft; (2) Pending; (3) Prepared; (4) Complete; (5) Cancelled. The business process design on the server side does not assume a federated model of e-commerce or e-business, and the processing of the order is done in a localized fashion. The created order is in a draft state (shopping cart) before moving to the pending state. The order object then moves from pending to complete once the order is accepted and completed by a sales agent on the server.

3.5. Disconnected business process model

The model of a regular business process (either on the client or the server) without the failure events and disconnection is a directed graph representing a state-machine, where each node as shown in figure 2 is a state of the process or an object (lifecycle), and each link is an action that is performed by the process when moving from state to state. Based upon the outcome of the action, the state of the process moves to the appropriate state as defined in the state-machine. Next we model the business process with and without disconnection and other failure conditions.

3.5.1. Disconnected process mechanism

The original business process P is defined as follows: $P = \{S, A, E, DT\}$. S is the set of states. Let $G(V, N)$ be a directed network graph of the state-machine, where V is the set of nodes in the graph, and N is the set of links in the graph. Each link corresponds to an action, defined in set A. An action is taken when an event occurs, and E is the set of events. An event could be a user input from the user-screen or an external input (e.g., arrival of a message from the server). DT is the data that the process accesses during its execution. A process engine, described later in more detail, runs an instance of a process as defined by the state-machine specification. Each state-machine is defined a collection of nodes and links, and is kept in a persistent state in the local database (e.g., collection of tables and graph objects). The process instance states are kept in a persistent local database as well. The disconnected business process is defined in the following fashion:

- $DP = \{S, A, E, F, FS, C, R, RA, DT\}$.

F is the set of failure events, C is the cost vector on resources, and R is the set of resources. FS is the set of disconnected and failure states, where failure events cause the process to move from one of the connected states to a failure state. The process is described a state-machine that consists of two kinds of nodes as shown in figure 2(b). Once in a failure state the process can move to the original non-failure state by performing certain recovery actions periodically as defined by the actions in set RA. These actions could be checking if the network connection is alive or if the corresponding system failure has been repaired. If it is, then the process can verify if the previous action completed on the server side. RA is the set of recovery procedures mapped to failure events. Similar issues and methodologies have been considered in [1,2,4,8] (e-business workflows for distributed transactions and messaging).

3.5.2. Disconnection and recovery

Using the above formal definition of a disconnected business process on the client, the process engine on the mobile device ensures that when a disconnection failure occurs due to a time-out or a network failure, the engine moves the process state to a disconnected state. Based on changing conditions, the process engine wakes up and checks to see if there are process instances that are in disconnected state (based on the business process state) and attempts to activate the process instances back to the original process states for final completion. The details of the recovery mechanism are in section 4.

The user of the mobile device can place multiple orders, each at different times. For each order process instance, the process engine checks to the see if the instance process state has completed, if not it proceeds to run the instance process towards completion. For example, a mobile user can create an order with 100 items in the morning and submit the order from the PDA, while this process is going through completion, a second order can be placed by the same user again for another 50 items (different from the previous order). Now

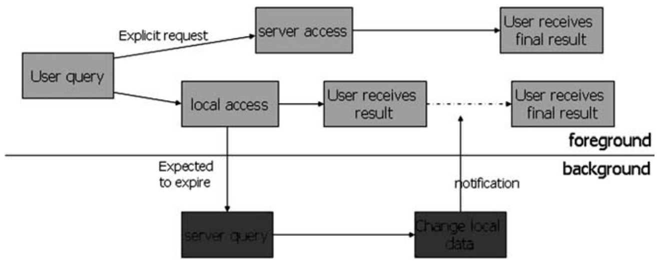


Figure 3. Transparency of local access and refresh from server.

there are two order process instances that have to be managed by the runtime engine of the middleware. Similarly, a different approach was done in [10,14,15], where more emphasis on recovery protocols is described.

3.6. Transparency

In order to support a basic level of transparency (as discussed in section 2), we adapted the process state-machine (figure 3) to include local and remote access to data. Note that this is a normal step, and can be used as a simple plug-in to the description of the process state-machine (figure 2), wherever needed. If the user requests access to local data, by default the request is satisfied primarily by the cached version of the data.

The use of timestamps can allow for more efficient updates. Timestamps and estimated expiration times (derived usually from the observed frequency of update) can be used not only for efficient synchronization, but also for transparency. It can be seen in figure 3 that if the local copy expires or is expected to expire, the local data will be refreshed before returning the answer if there is a connection available to the server.

A completely transparent system can be misleading to the user. There are cases where the user's feedback is essential for good performance. Consider a case where the user formulates transactions based on data that the system knows are expired. Due to locking mechanisms that support serializable transactions, the inconsistency is not visible to the user. We believe that the user should be involved in a reasonable amount of the decision making, such as the receipt of notifications when it is advantageous to notify and make local rather than distributed rollbacks. In some cases, the user can activate the system to do the recovery process.

3.7. Efficiency

As we already discussed, it is important for the local cache to be as up-to-date with respect to the server as possible. The question is how to allow for most user requests to be satisfied locally, and reduce communication costs. Several prefetching strategies have been proposed for caches, but most of them work well under assumptions such as locality of access, which may not be the case in mobile computing. For example, when the e-commerce server is accessed by a large number of clients, it cannot store its data such that all clients can benefit from the principle of locality.

Due to the high cost of aborting distributed transactions, and the lower cost of batch updating, pre-fetching mechanisms are essential. Update frequency information can be incorporated into these mechanisms to further reduce the cost of communication. Due to transparency requirements, the local cache (also called local DB) of the mobile client should be refreshed as a secondary process while explicit user operations are in the foreground (figure 3). These two layers should not interfere and maintain a consistent state of the local database and efficient transaction execution. Main challenges are the following:

- Find good refresh mechanisms that work for mobile devices and involve almost no support from the server. For this, we considered a new class of algorithms that fetch data based on inferences from the process state-machine.
- Scheduling of background refresh processed so that their actions do not have a significant negative impact on user actions, and do not impose a high communication price. Also, user requests should be given priority in a client to server message queue.
- Ensure a consistent view of the data accessed by one user, within one process.

3.8. Synchronization

Our system involves minimal support from the server. The support of multiple users accessing one device, and multiple devices accessed by one user, complicate the maintenance of data and process states as part of profiles. Therefore, in our design of a mobile e-commerce client, we assumed no replication of the local cache on the server.

3.9. Support for advanced transaction models

Traditional transaction models are not meant to support distributed actions involving mobile devices. By contrast to computing devices that have persistent network connection, mobile devices can be disconnected frequently and for long periods of time. For implementing a transaction model, this fact has several implications:

- A client cannot request the locking of data items on the server. Since there are no guarantees when a mobile device will re-connect to the server, it is hard to define reasonable timeouts on locks over server data.
- Distributed transactions can have a very high abort cost. When such a transaction is aborted, not only the local data has to be refreshed and the transaction resubmitted, but, due to disconnections, the mobile device may be notified about the abort after a long period of time.

One possible solution is to break down distributed transactions into sub-transactions, based on requirements of the business process states. This would ensure that, if possible, part of the transaction can be committed even if a sub-transaction is aborted.

4. Architecture

In this section, we present some of the design criteria for developing a "light weight" client middleware for enabling e-business and computing applications to run in all kinds of potential modes of the device and the user.

4.1. Design criteria

There are many design criteria to consider when developing and deploying and end-to-end solution for mobile commerce on clients and servers. The clients could be PDAs, PocketPCs, wireless laptops and other mobile devices. The criteria for client-side middleware are as follows:

- *Disconnection mechanisms.* Handling disconnection, failures and resource constraints (such as low memory or low battery or network bandwidth) and providing the application user to access and perform business activities in seamless fashion.
- *Synchronization mechanisms.* Providing mechanisms to synchronize local and remote data and processes for business applications.
- *Event management* to handle all kinds of system failures, network disconnections, low battery, resource limits and others.
- *Self-management.* A process engine to drive the business processes on the client side for various object instances and process instances. For example, in the business scenario in section 3, a process engine drives the order process based on the defined state-machine. The process engine is needed to handle recovery procedures for each active business process when network connection is back-up or a resource is back in operation.
- *Authentication and session management.* The client and server should handle mobile sessions similar to regular web-browsers for some applications. Session management is done through cookies, which have to be set by the servers in the responses, and conforming to the URL-encoded schemes. The clients have to maintain session information for interaction with the servers.
- *Run-time system.* A run-time environment to execute business processes, handle business objects, handle failures, and persistence is needed to run business applications.
- *Member profiles.* With mobile commerce evolving at a rapid pace, some of the participants at the e-commerce site (eMarketplace or sell-side server) will have more than one device, and they would like to use more than one-device to place-orders, and receive notifications.

4.2. Architecture on the client device

An architecture illustrating the device integration with the server is shown in figure 3, where mobile devices are connected via protocols such as Webservices [5,20] and others to the supplier and distributor websites. Webservices in the coming years are going to be crucial for intar-enterprise and inter-enterprise integration. Our architecture is based on web-service components such as UDDI, SOAP and WSDL [5,20] for enablement of server-side services for mobile clients to access. For the scope of this paper, we will not describe the Webservices enabled on the server-side for catalog retrieval, member authentication and order-placement. As of now Web-services are still undergoing issues on security and authentication [5,20].

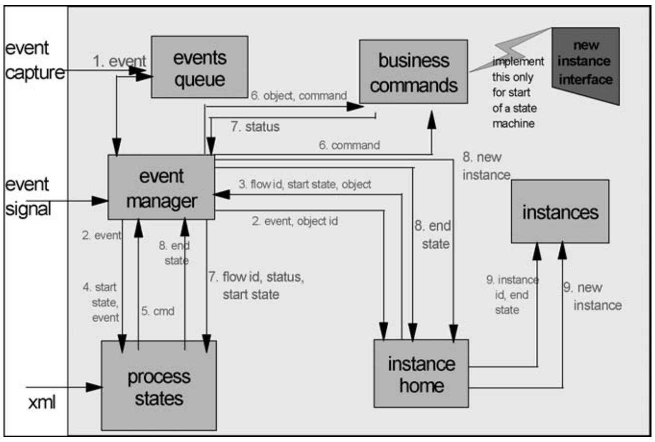


Figure 4. State-machine and event management.

The figure shows three main components that form the core of the framework.

Persistence and messaging. The first component is the database system, which is a reduced functionality embedded relational database system [6]. The second component is the messaging component to handle in-bound and out-bound messaging in a reliable fashion.

Run-time engine. The third, and the most important component is the run-time of the e-business framework. This run-time engine handles the following: (1) persistence of business objects in the local embedded relational database; (2) a process engine to handle all business processes, interactions and all kinds of events (including failure events); (3) a set of recovery mechanisms which are run "automatically" to enable the application process state to move from disconnected to connected states and final completion of the process.

Self-management. The run-time engine through event management can handle all kinds of failure events as long as they are captured and put in an event queue. In addition, the engine can self-recover the processes that are in disconnected or failure states by performing simple recovery procedures.

4.3. Client side engine for business processes

In figure 4, we illustrate the process engine, which is part of the client side middleware that runs a business process to completion. The core sub-components are an event mechanism, a process engine, a queue for all kinds of events, an instance factory for process instances.

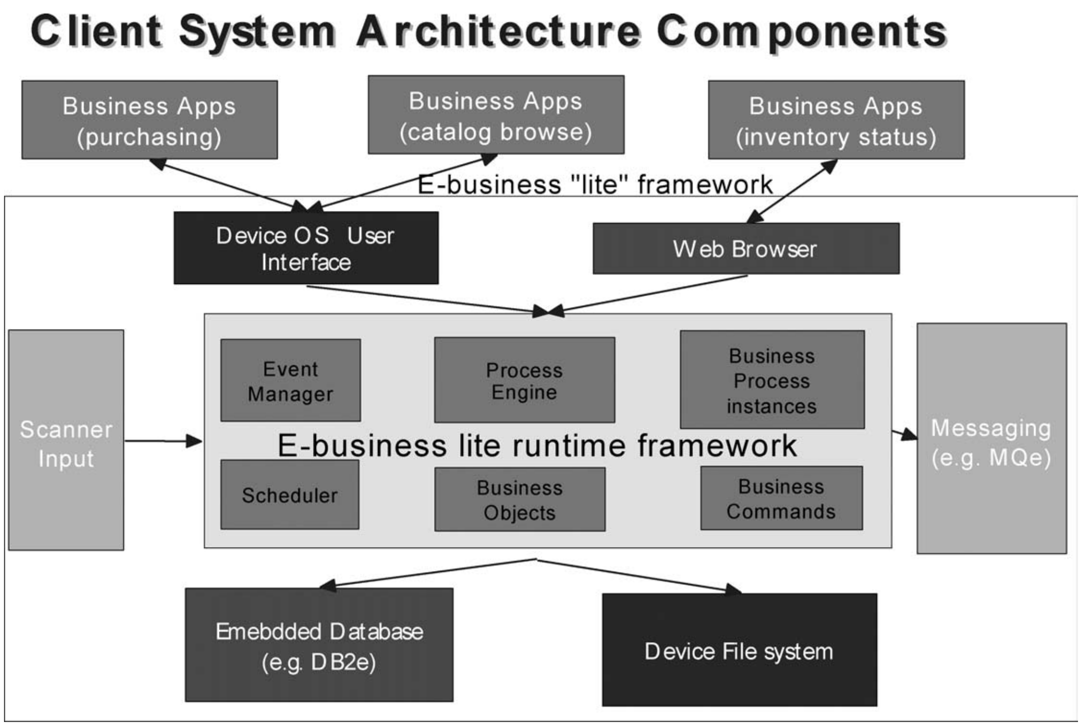

Client System Architecture Components
Business Apps (purchasing)
Business Apps (catalog browse)
Business Apps (inventory status)
E-business "lite" framework
Device OS User Interface
Web Browser
Scanner Input
Event Manager
Process Engine
Business Process instances
E-business lite runtime framework
Scheduler
Business Objects
Business Commands
Messaging (e.g. MQe)
Emebdded Database (e.g. DB2e)
Device File system

Figure 5. Client side architecture components.

Every event is handled by the event manager, which is managed by the process engine. The events are tagged with the instance id or the process id to specific which event corresponds to which process instance. Every event executes a business command or a recovery action (as described in figures 2(a) and (b)).

4.4. Programming environment

The programming environment (see figure 5) consists of many core components that enable an application programmer to rapidly develop and deploy applications. The core programming layers and components are the following:

- A user interface layer for enabling applications to handle user input and corresponding actions for every user input and action.
- Business objects such as products, member profile, shopping cart, orders, session and others. A persistence framework to enable persistence of business objects in an embedded relational database such as DB2e [6].
- A collection of commerce (or e-business) commands which run the business logic for local catalog reads and writes, adding items to local shopping cart and placing orders locally and submitting orders to the server. The commerce commands are simple java objects with transaction capabilities to read and write to the local relational database. Commands form an integral part of the business logic for a process.
- A run-time engine to enable access to process instances and process states, and an event manager to enable writing to an event queue, and reading from an event queue.

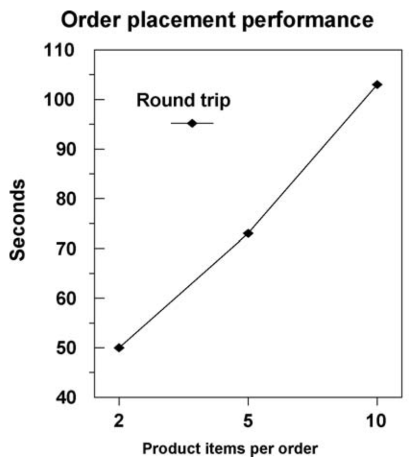
5. Configurations and deployment performance studies

Each mobile device has a local database systems (DB2e from IBM), a messaging component (MQe from IBM [6]), a Java run-time engine (J9 JVM from IBM [6]), the J2ME framework, and a collection of XML and User Interface libraries. In the configuration, the mobile devices (PALM, WinCE and Windows 2000 (laptops) based devices) are connected to the Internet via wireless Ethernet cards enabled with 802.11b protocols. A base-station provides the link between the wireless device and the wired LAN. In this configuration, a request from a mobile client is sent directly via reliable messaging or direct synchronous TCP/IP connections to a commerce server (which is an IBM product WCBE [6] for e-business applications). The total footprint which inlcuded the client side middleware, embedded DB2, XML parsers and the purchasing application was around 1.8 Mbytes, enough to run smoothly on PDAs and PocketPCs.

5.1. Performance results

The following experiments were done to investigate the performance of the application middleware, J2ME and the application itself. The applications we studied were as follows:

- *Catalog retrieve operation.* A user of mobile device retrieves a part of the server catalog onto the mobile device for browsing the prices and availability of product items.
- *Order placement operation from the mobile device.* The user of the mobile device creates purchase orders on the device. The orders once submitted by the user are sent across to the commerce server for processing.



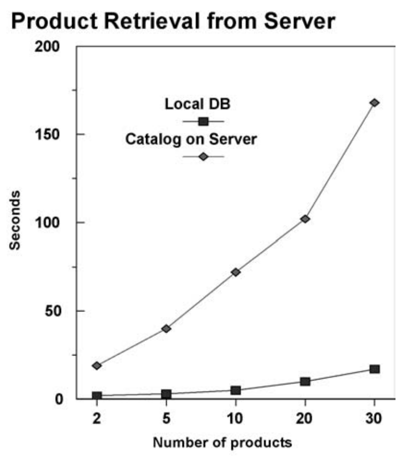


Figure 6. Performance of order placement.

In figure 6, the total response time for placing an order is around 50 seconds for a 2 item order, and 105 seconds for a 10 item order. This is fairly reasonable when compared to placing an order through a regular web-browser from a desktop machine. We noticed that most (80%) of the delay is on the server side, and not much on the client side for smaller number of order items in an order.

Once the order is placed or submitted by the users, the following interactions occur automatically between the client and the server: (1) the client first issues a login request to authenticate the user; (2) the client sends the shopping cart to the server for creating a draft copy of an order on the server; (3) a prepare order is sent to the server for pre-completion checks; (4) a complete order is sent for fulfillment and sends back a confirmation to the client. In some cases the response for order confirmation might arrive later due to a manual process in the order acceptance.

The catalog retrieval operation took about 168 seconds delay for getting 30 products from the server. Each interaction involved authentication and other overheads. The performance numbers are high due to server delays, network delays and processing delays on the client. The breakdown is as follows. The local delays are due to parsing of responses from the server. What we found is that local processing delays are 10–20% of the overall delays, depending on the size of the catalog or order placed. Network delays in a wireless-LAN environment are very small. The server-side processing delays occupy 70–80% of the total response-times. The delays for every transaction are also due to authentication round-trip messages between the client and the server.

The J2ME framework is still undergoing changes, and this will improve in the coming years. However, the interesting result is the contrast between accessing the local catalog on the client relational database and accessing the catalog on the commerce server. The local access to the 30 product items is just under 3 seconds and with faster clients (e.g., iPAQs), this is sub-seconds. This demonstrates that running e-business applications with such high performance local databases

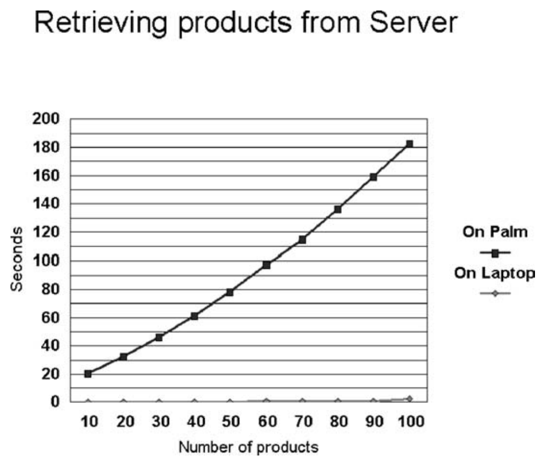


Figure 7. Comparison of retrieval performance between devices.

is beneficial. With the order placement showing considerable performance, the user can perform purchasing activities in a transparent fashion with a large catalog of items on the local database.

5.2. Performance on multiple devices and operating systems

In this section, we compare the performance results for catalog retrieval and order placement on PDAs and Wireless Laptops with the same software base, embedded database and applications. The middleware is independent of the device because of the underlying J2ME framework provides a good level of transparency with well-defined APIs that hide the device specifics.

In figure 7, we compare the performance of the same purchasing application and same middleware on the two devices. A 100 catalog products are retrieved from a server by the application running on the two devices. For the wireless laptop, the processing speed of the incoming XML file is much faster, and this contributes to the difference in the response time. For the Palm device (32 Mb, 32 Mips), the response time for retrieval is 200 seconds for 100 products. With better XML processing and webservices technology on the Palm

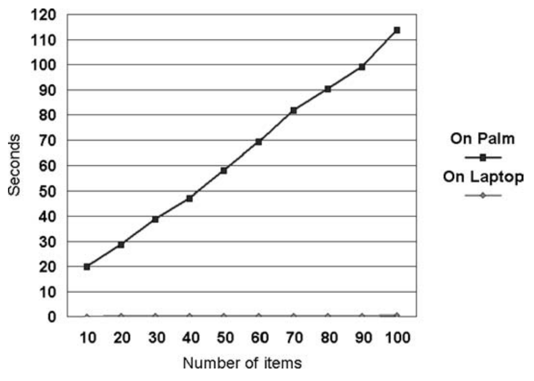


Figure 8. Order placement to remote server.

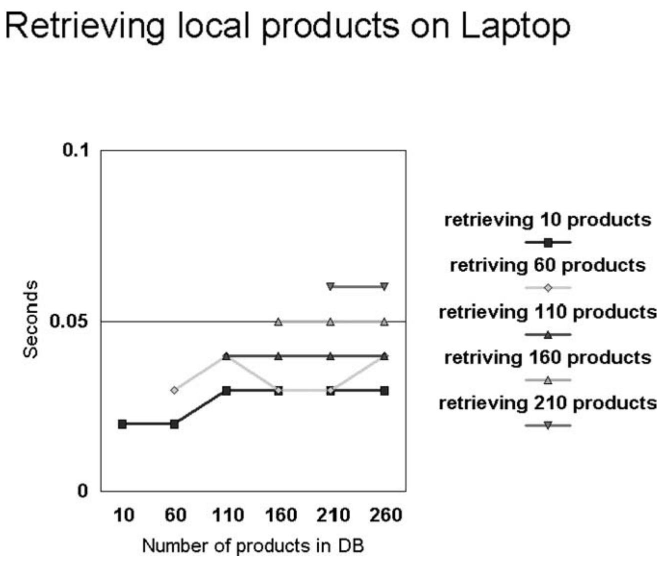


Figure 9. Response time for catalog retrieval from local DB on laptop.

client, the response time can be further reduced. In figure 7, we show the catalog retrieval on a laptop (IBM ThinkPad), which has 512 Mbytes of RAM, and the processor speed was 800 MHz. The catalog retrieval on a laptop takes sub-seconds for 100 product items.

In figure 8, we compare the response time for order placement from the devices to the server. The response time is the roundtrip from the client to server for order placement (including the order acknowledgement). The mobile user places an order with a certain number of order items from the shopping cart application on the client.

The response time for order placement from the device to the server is higher for the Palm device (capabilities of 32 Mbytes and a processor speed of 32 Mips). The response time increases almost linearly with the number of items in the order that is placed by the mobile user on the Palm device. For the Palm device the total response time for 100 order items is 118 seconds. For the wireless laptop, the response time is less then a second. The difference is mainly due to the processing speed of the devices.

In figure 9, we show the performance of catalog retrieval on a laptop (with 512 Mbytes of memory and 800 Mips). The retrieval of 200 items is about 0.1 seconds for local access, and about 2 seconds for 100 items from a remote server. Wireless laptops are becoming as powerful as small servers, and PocketPCs are reaching the capabilities of current laptops. From a cost performance point of view, we illustrate that the devices can provide excellent capability for business transactions such as order placement for hundreds of order items.

6. Summary and conclusion

In this paper, we presented disconnected business process models and mechanisms for enabling transparent mobile e-business applications with local access and synchronization between the mobile client and server. We presented a detailed architecture and a programming framework for end-to-end mobile e-business applications. This architecture was designed and implemented, and is currently undergoing trials with various e-business applications. The performance numbers demonstrate that the users who wish to browse prices and availability of hundreds of catalog entries can do so at leisure while being mobile by doing most of the commerce transactions on the client, and then placing an order to the server when connected.

We also compared the performance of the middleware and a purchasing application on devices with different capabilities. We show that for retrieving hundreds of catalog items or placing large orders from the client, the performance is very reasonable on smaller devices such as PDAs and PocketPCs. For wireless laptops, the response time performance, as expected, is excellent. With increasing device capabilities and improved client side middleware and software frameworks, mobile devices will increasingly become attractive for consumers and businesses for daily business activities, and will in the near future redefine the way computing is done by businesses and consumers.

References

[1] G. Alonso et al., Exotica/FMDC: A workflow management system for mobile and disconnected clients, Journal of Distributed and Parallel Databases (1996).

[2] M.H. Dunham et al., A mobile transaction model that captures both the data and movement behavior, Mobile Networks and Applications 2(2) (1997) 149–162.

[3] Durlacher Research Ltd., Mobile Commerce Report (February 2000).

[4] D. Georgakopoulos et al., An overview of workflow management: From process modeling to workflow automation infrastructure, Journal of Parallel and Distributed Systems (1995).

[5] K. Gottschalk, S. Graham, H. Kreger and J. Snell, Introduction to Web services architecture, IBM Systems Journal 41(2) (2002).

[6] IBM, Embedded database system: DB2e and Websphere commerce, `www.ibm.com`

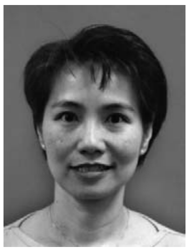
[7] A. Jhingran, Moving up the food chain: Supporting e-commerce applications on databases, SIGMOD Record 29(4) (2000) 50–54.

[8] P. Muth et al., From centralized workflow specification to distributed workflow execution, Journal of Intelligent Information Systems 10(2) (1998) 159–184.

[9] B. Noble et al., Agile application-aware adaptation for mobility, in: *Symposium on Operating Systems Principles (SOSP)* (1997) pp. 276–287.

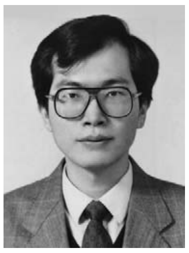
[10] D.K. Pradhan et al., Recoverable mobile environment: design and trade-off analysis, in: *International Symposium on Fault-Tolerant Computing (FTCS)* (1996) pp. 16–25.
[11] P. Ram et al., Distributed Transactions in practice, SIGMOD Record (1999).
[12] J. Sairamesh et al., A platform for sell-side private exchanges, IBM Systems Journal 41(2) (2002).
[13] J. Sairamesh et al., Wireless B2B trading, in: *Proceedings of the 1st ACM Workshop on Mobile Commerce* (conjunction with *Mobicom*, 2001).
[14] M. Satyanarayanan, Mobile information access, IEEE Personal Communications (February 1996).
[15] M. Satyanarayanan, Fundamental challenges in mobile computing, in: *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, Philadelphia (May 1996).
[16] SyncML Consortia, `www.syncml.org`
[17] D.B. Terry et al., Managing update conflicts in Bayou, a weakly connected replicated storage system, in: *Symposium on Operating Systems Principles (SOSP)* (1995) pp. 172–183.
[18] P. Timmers, *Electronic Commerce: Strategies and Models for B2B Trading* (Wiley, 1999).
[19] A. Tsalgatidou et al., Challenges in mobile electronic commerce, in: *Proceedings of the IeC 2000, 3rd Internat. Conf. on Innovation through E-Commerce*, Manchester, UK, November 14–16 (2000).
[20] Webservices Architecture-Working Draft, W3C-Architecture-WorkingGroup, `www.w3c.org`

Jakka Sairamesh is a project leader and senior research staff member at IBM Watson Research, Hawthorne, New York. He obtained his M.S. and M.Phil. from Columbia University in 1992, and Ph.D. from Columbia University in 1995. Since then has been working with IBM Research on e-business middleware, auctions and trading systems, mobile e-business, information economies, and decentralized computing systems. He has published numerous papers on trading systems, e-business platforms, market-based control, and information economies. He was one of the architects from Research for IBM's e-business products such as Websphere Marketplace Edition and Websphere Business Edition, and recently helped drive the vision, strategy and architecture for Business Portals for Value-chain management and collaboration. He currently leads projects in the areas of dynamic e-business solutions, early warning systems, large-scale mobile and peer-to-peer computing, dynamic resource management in Grids, and Internet economics.
E-mail: jramesh@us.ibm.com


SweeFen Goh is a software engineer working in the e-commerce department at the IBM Watson Research Center. She received her B.S. and M.S. degrees in computer science, with a minor in mathematics, from Kent State University, OH. She has more than ten years of professional experience in both Industries and Commercial Research. She has held leading position in software development, solution architecture and project management. Her current research interests are in the areas of mobile commerce, data synchronization in multi-channels application environment and distributed authorization.

Ioana Stanoi is a research staff member at IBM T.J. Watson. Her research interests include mobile computing, data synchronization in distributed systems, data discovery and retrieval for P2P systems, and storage and retrieval methods for structured and semi-structured data. Ioana obtained her Ph.D. degree in computer science at the University of California at Santa Barbara in December of 2000, after receiving a Bachelors of Science in computer science and a Bachelors of Arts in physics at the same University.

Sriram Padmanabhan was a manager at the IBM Watson Research Center which he joined in 1992. Dr. Padmanabhan currently has moved to a new position in IBM, San Jose. He was an active member of the DB2 Parallel Edition research and development effort and had contributed to several architectural aspects of the system and significantly to the query optimization and parallel query execution areas. Besides parallel databases, he is interested in object-oriented and distributed databases, information retrieval, massive data-storage applications, and parallel architectures and algorithms. Dr. Padamanabhan received his Ph.D. and M.S. degrees in computer science from the University of Michigan, Ann Arbor, in 1992 and 1990, respectively, and his B.Tech. degree from the Indian Institute of Technology, Madras, India in 1986.


Chung-Sheng Li received his B.S.E.E. degree from National Taiwan University, Taiwan, R.O.C., in 1984, and the M.S. and Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley in 1989 and 1991, respectively. He has joined the Computer Science division of IBM T.J. Watson Research Center as a research staff member since September 1991, manages the Image Information System Group from 1996 to 1999, the Data Management Department between 1999 and 2000, and assumes the senior manager position for the E-Commerce and Data Management Department since June 2000. He has initiated and co-initiated several research programs in IBM on fast tunable receiver for all-optical networks, content-based retrieval in the compressed domain for large image/video databases, federated digital libraries, and bio-surveillance. He is currently the principal investigator for a environmentally related digital library funded by NASA and a bio-surveillance project funded by DARPA.

Dr. Li has received a Corporate Environmental Affairs Excellence Award in 2003 for his leadership in the EpiSPIRE project, an Outstanding Innovation Award in 2000 for his leadership and major contribution to the IBM/NASA digital library project, and a Research Division award from IBM in 2001 and 1995 for his major contribution to the transcoding for universal access by pervasive devices and tunable receiver design for WDMA, and numerous invention and patent application awards. He was an associate editor for the IEEE Transactions on Multimedia and currently an associate editor for Journal of Computer Vision and Image Understanding. He is also part of the Distinquished Lecturer Program of the IEEE Circuit and System Society (2002–2003). He has authored or coauthored more than 120 journal and conference papers and received one of the best paper awards from the IEEE International Conference on Computer Design in 1992, and the best paper award from the IEEE Transactions on Multimedia during 2003. He is an IEEE Fellow in the Circuit and System Society, the Laser Electro-Optic Society, the Communication Society, and the Computer Society.