Gateway:A Message Hub with Store-and-forward Messaging in Mobile Networks

Eiko Yoneki and Jean Bacon

University of Cambridge Computer Laboratory,
William Gates Building, J J Thomson Avenue
Cambridge CB3 0FD, UK
{*Eiko.Yoneki, Jean.Bacon*}*@cl.cam.ac.uk*

Abstract

To obtain good performance in messaging over mobile networks, we have developed a Gateway. **Gateway** *is a message hub that transmits information using store-and-forward messaging and provides powerful optimization and data transformation. The* **SmartCaching** *component provides generic caching in an N-tier architecture, an essential function of Gateway. Gateway can be integrated into Pronto, a middleware system for mobile applications with messaging as a basis [15]. Pronto then offers: 1) a lightweight client of Message Oriented Middleware (MOM) based on Java Message Service(JMS), 2) Gateway for reliable and efficient transmission between mobile devices and a server, and 3) Serverless JMS based on IP multicast. Integration of Gateway within Pronto provides a solution for mobile application-specific problems such as resource constraints, network characteristics, and data optimization.*

1. Introduction

A large-scale distributed system must offer load sharing and reduction for good performance. Computing devices are increasingly mobile at the client end and the diversity of clients and networks creates complex requirements for mobile/wireless based applications. The communications service provided by middleware is especially important for integrating such hybrid environments into coherent distributed systems. The characteristics of mobile computing [4] and wireless networks, and the corresponding requirements on middleware, are shown below:

- Mobile devices have small ROM/RAM footprints and low usage of CPU cycles and power. A middleware client library should have a small memory footprint.
- Wireless networks have become increasingly packet-oriented. With a packet- oriented bearer such as GPRS (General Packet Radio Service) or UMTS (Universal Mobile Telecommunications System), users typically pay only for the time they communicate data. Reducing data size for transmission is crucial.
- Because of low bandwidth, high latency, and frequent disconnections, a middleware should provide an interface to applications that allows the maintenance of communication during the disconnect operation. Dependable caching is essential.
- A data source can be interpreted in different formats and semantics depending on the specifications of mobile devices and wireless networks. Semantic transcoding technology [10] should give advantages for efficient data flow.
- There are various bearers such as 2G, 2.5G, 3G, Bluetooth, and IEEE 802.11 and many devices are non-programmable. A middleware needs to offer an interface which provides a communication abstraction.
- There are different operating systems on mobile devices and a multi- platform middleware should be implemented in a platform independent language.

The architecture of a distributed system at this level needs careful consideration, and it is essential to provide the core function for such a system as semantics-based middleware. We have developed **Pronto** [15], a middleware system for mixed environments which include mobile applications. The basis of Pronto is a Message Oriented Middleware (MOM) [6] based on Java Message Service (JMS) [12] in both centralized and decentralized forms. JMS works well in an environment where network connections sometimes break, and the available bandwidth can vary within a short time. JMS provides the architecture for MOM; it encourages loose coupling between message producers and message consumers with a high degree of anonymity, thus removing static dependencies in the distributed environment. The decentralized form, Serverless JMS, uses IP multicast and performs best over ad-hoc networks, and also for high-speed transmission of a large number of messages to distribute the workload of a server to several servers. Figure 1 shows a system overview, illustrating different deployments of Pronto. In Pronto the client library, MobileJMS Client, is optimized as a mobile-specific JMS client for con-

Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)
0-7695-1921-0/03 $17.00 © 2003 IEEE

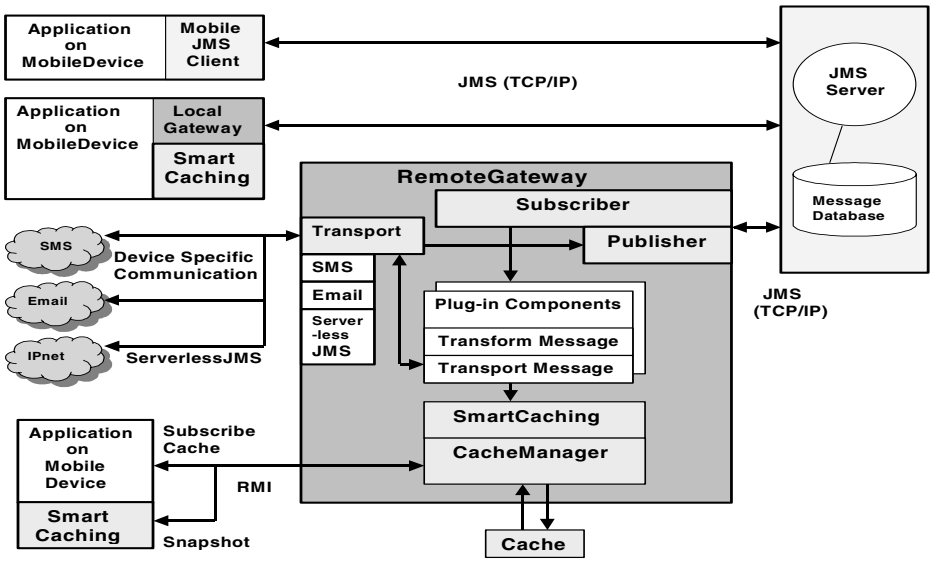

Figure 1: System Overview with Gateway and SmartCaching in Pronto

strained mobile devices. We have also developed an intelligent **Gateway** that resides between mobile applications and servers. Gateway is a message controller which gives reliable transmission and efficiency, taking advantage of plug-in components for caching, device specific transport, and message transformation. The **SmartCaching** component is designed to provide generic caching with subscribe and snapshot services. It is a central function for message storage in Gateway. Gateway and SmartCaching are key technologies for improving messaging among mixed mobile-tier environments in dynamic connectivity scenarios. Gateway can be well integrated with JMS in Pronto and Pronto allows the building of a dynamic, reliable, and flexible system with significant performance improvements. This paper presents the design and implementation of Gateway and SmartCaching.

2. Gateway

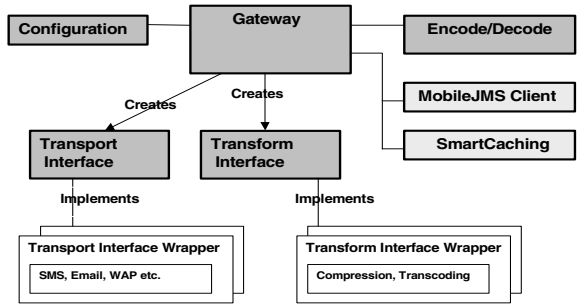

Figure 2: Gateway Components

In messaging, there has been an effort to support a mobile-tier structure by adding an edge server to manage mobile devices. Gateway takes a different approach by providing multiple message hubs for transmitting information using store-and-forward messaging, thus giving more powerful optimization of data reduction and transformation. This makes it possible to construct a distributed messaging system over JMS servers. The main Gateway components are shown in Figure 2. Gateway is designed as a framework to perform plug-in functions for which two interfaces are defined:

- **Transport**: an interface for mobile device transport
- **Transform**: an interface for message transformation

The plug-in functions should follow these interface definitions and their internal details are not discussed further. Gateway initially creates *Transport* and *Transform* objects, according to the configuration; a sample is shown in Figure 3. It contains the class names that implement the transport and transform interface, and the target topic names indicate the message groups to be transformed. The Encode-Decode component carries out the message transformation as defined in the configuration. Gateway is also a MobileJMS client and can reside in a mobile device. SmartCaching is used to store messages. The implementation is 100% in Java.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <cache      state="1" size=100 />
    <transport  state="1">
        <bearer name="sms"   classname="transport.SMSTransport" topic="Stock"/>
        <bearer name="email" classname="transport.EmailTransport"/>
            . . .
    </transport>
    <transform  state="1">
        <transformer name="compress" classname="compression.Compression"/>
        <transformer name="image" classname="transform.Image" topic="Grey"/>
        <transformer name="audio" classname="transform.Audio" topic="News"/>
                . . .
    </transform>
```

Figure 3: Configuration for Plug-In Components

2.1 Gateway Operation

Gateway distributes messages to other Gateways and to applications. Many gateways can be used, as appropriate for the network environment and client characteristics. Messages commonly contain requests to and responses from data sources which must identify the needs of requesters. Cascading Gateways provide distributed filtering to minimize network traffic and the message transformation process. As the distance from the source increases the data becomes more localized. The operation flow of Gateway is shown in Figure 1.

A part of Gateway acts as a publisher and subscriber using a MobileJMS Client to serve as a proxy of a message controller. Another part performs a series of message transformations on subscribed and published messages. Gateway defines a *Transport* interface to perform the device-specific communication and provides a store-and-forward communication model that offers load sharing and load reduction for good performance.

2.2 Local and Remote Gateway

Gateway itself is defined as an interface, with two implementations **LocalGateway** and **RemoteGateway**. LocalGateway can run as a separate thread or within the application and performs caching and transcoding through plugged-in components. RemoteGateway is currently implemented as a RMI [14] *UnicastRemoteObject* and can run as a separate process. Mobile devices can take advantage of both Gateways depending on the application. Deployment possibilities are shown in Figure 1.

2.3 Plug-In Components

Caching, compression, and semantic transcoding are good candidates to reduce data size and network traffic. Security (encrypting/decrypting data) functions can also be plugged in. Semantic transcoding offers more than a simple data reduction. The information itself is made more abstract (to provide compaction), and the data should be evaluated whenever necessary. In a mobile/wireless environment, a reduction of data size on the network dramatically increases performance, and the concept of semantic transcoding is important. Here, the data are linked to an annotation. Annotations [10] can be text corresponding to a video clip, a summary of a document, or a linguistic annotation of the content for voice synthesis or greyscale/downsized/low-resolution image data.

2.4 Non-Programmable Transport

Transport is an interface to manage non-programmable devices. The registration of a *Transport* interface to Gateway activates a subscription to a JMS server on the specified topic. Messages that are delivered from the server to Gateway will be forwarded to *Transport*, which looks up the device and session lists and sends messages accordingly. Messages published via *Transport* are forwarded to a JMS server. Figure 4 shows the control flow of the *Transport* interface.

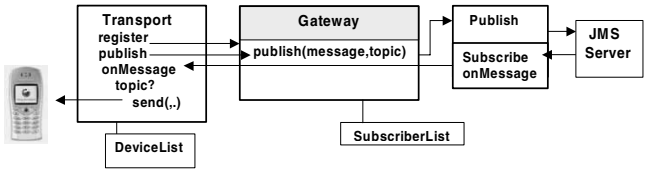

Figure 4: Non-Programmable Device and Gateway

2.5 SmartCaching

Gateway embeds SmartCaching to store JMS messages. The JMS durable subscription is used to receive messages continuously even while the client is out of contact. On request types such as subscribe or snapshot, Gateway stores subscribed messages accordingly. See Section 3 for generic SmartCaching.

2.6 Disconnect Operation

Most work dealing with the disconnectedness of computing devices revolves around data replication and synchronization. The following approaches are designed for disconnected operation in Gateway:

Durable subscription via MobileJMS Client: Durable subscription is defined in the JMS API. Non-durable subscriptions last for the lifetime of the subscriber object. The client will only see the published messages while the subscriber is active. A subscriber can, optionally, be durable by registering a durable subscription with a unique identity.

Gateway Cache: Gateway maintains its cache even if applications are inactive. Applications can use the Gateway cache after regaining connection; they can use the pull, subscribe, and snapshot operations of SmartCaching as appropriate. For example, an application may spawn a snapshot request that synchronizes the on-device messages when the application is reconnected.

3. SmartCaching

Caching is essential for performance improvement by reducing network traffic and improving latency. The cached data can be raw or processed and stored for reuse, thus avoiding revisiting the source and passing the data through the chain of reformatting and representation. SmartCaching, an intelligent cache function, supports multi-tiered applications across platforms and devices. It currently implements basic functions, while persistent caching, cache validation, synchronization, and coherency management are beyond the scope of this study. In SmartCaching, cached data is decoupled from the data source, and cached data can be made active or up-to-date by CacheHandler,

which is responsible for updating the cache. For example, Gateway is a CacheHandler, and it uses SmartCaching to store subscribed messages. Key functions to clients are the **Pull, Subscribe** and **Snapshot** services. The Subscribe service provides asynchronous notification of cached data to client applications, and applications do not need to request to pull the data that have already been requested. Using the Subscribe service client applications may be event-driven and active. This simple change has a major impact on performance and on the design of the applications. Snapshot provides a specified period that can be used by the mobile application to obtain the last cache image after disconnection. *CacheManager* is the main component in SmartCaching. It creates objects and manages requests and responses to the requesters. Cache is an object that contains a key and the actual caching object, kept as a linked list. The *Cache* object contains the expiration date, and the CacheManager will remove expired objects. Alternatively, the Cache object can be removed once it is delivered to the subscriber. The three main functions above operate in response to requests from *CacheManager*.

Pull: An application requests a cache synchronously.

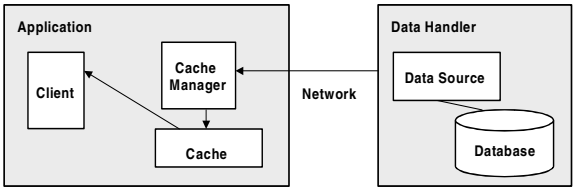

Figure 5: SmartCaching: Pull

Subscribe: An application requests a cache update notification to a cache handler, which notifies the application after the cache is updated.

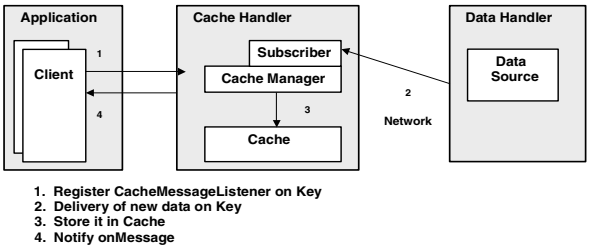

Figure 6: SmartCaching: Subscribe Cache

Snapshot: When data are delivered piecemeal to applications in a time sequence, clients should be able to reconstruct the latest view of the information. This can be achieved by obtaining all data from the data source or by retaining the last image in a shared cache. The second option corresponds to the Snapshot service. If the data source sends messages via minimal delta information, caching updates existing data, applying only the delta information. Snapshot needs to know when the baseline starts. Each time a new message is received, the Snapshot rule is applied and persists the data in the cache.

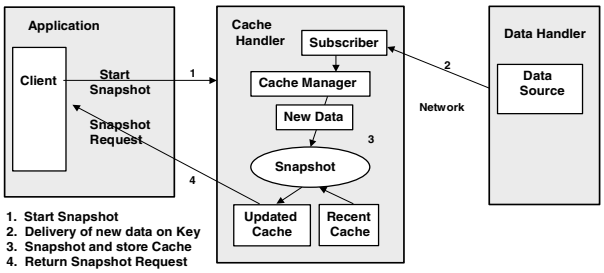

Figure 7: SmartCaching: Snapshot Data Flow

The Snapshot rules can be provided by an application. If a client requests Snapshot, it will receive the latest data only. It is the responsibility of the client application that made the Snapshot request to retain all data, and, after the snapshot's arrival, to apply the data to bring that snapshot up-to-date. Gateway uses Snapshot continuously to receive messages, even while the client is out of contact, and it passes them on when the client reconnects, upon the Snapshot request. Meanwhile the client is able to continue to operate using its own local cache to satisfy the requests as far as possible. After restoring communication, only the last image of the cache needs to be updated. This can reduce the need for reconnection by skipping all intermediate data. The event notification mechanism allows the notification to applications of later changes in the underlying cached data. When Snapshot is on, cache update notification is done only when the last image changes. The data flow of Snapshot is shown in Figure 7. Figure 8 shows examples of Snapshot rules. In the first example, a message contains a delta value from the base, and the rule is simply to carry out an arithmetic operation. In the second example, a message is added to the tail of the previous one.

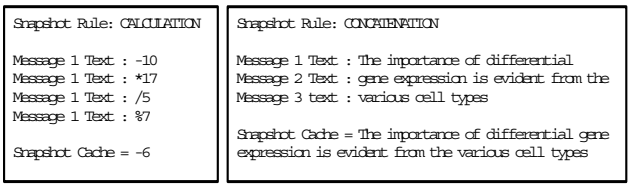

Figure 8: Examples of Snapshot Rules

4. Evaluation

An example application and some samples from the benchmark test are shown below.

4.1 Video Data Publishing in a Time Sequence

Figure 9 shows an example system with Gateway and SmartCaching. A video camera takes 15-second shots every

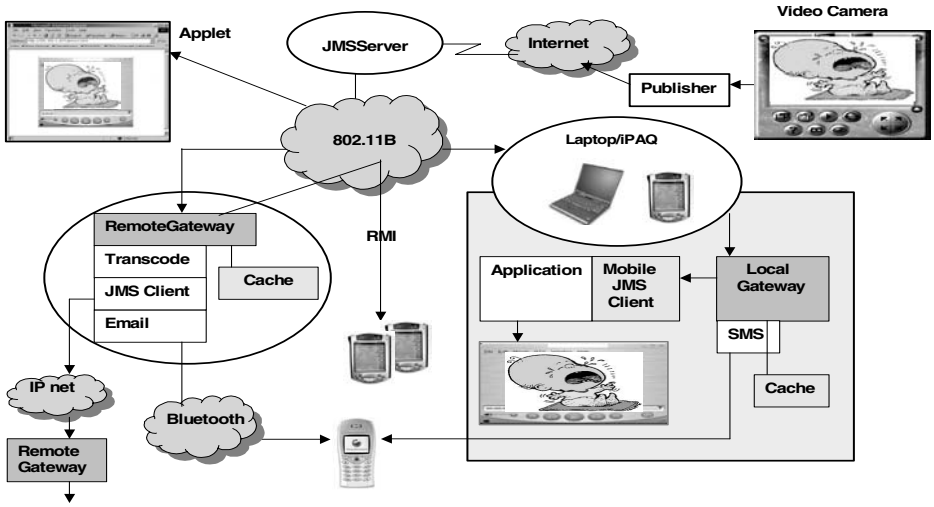

Figure 9: Video Data Publishing in a Time Sequence over 802.11B Network

minute, and data are published on a specific topic. The Laptop/iPAQ device is moving, leading to occasional disconnections. LocalGateway, running on the device, is using the durable subscription and all the published data are stored in the cache. A plugged-in SMS component sends out an SMS message after transcoding. At the same time, RemoteGateway subscribes to the same topic and emails to the phone after transcoding. Several iPAQs subscribe to cached data from RemoteGateway, and all of them get the data via RMI. This example demonstrates that published video data are distributed to the mobile devices with efficient data optimization.

4.2 Benchmark Test over 802.11B

Caching: This test focuses on the performance of SmartCaching in RemoteGateway. 50 KB x 20 BytesMessages are published and RemoteGateway subscribes to and caches them. Each subscriber listens to the cache update notification from RemoteGateway. Sharing the cache among subscribers reduces traffic overhead from individual subscriber's requests. Thus, the result shows better performance with more than one subscriber, and an increase in the number of subscribers does not have significant impact on performance.

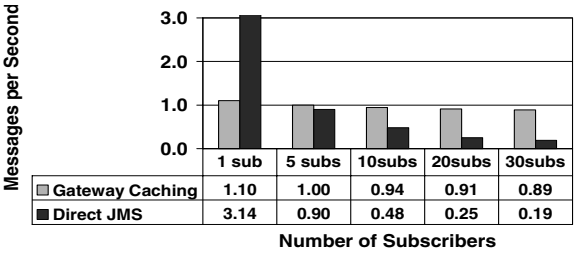

Figure 10: Performance Improvement by Caching

Text/Audio Semantic Transcoding: This test focuses on performance improvements by semantic transcoding. 1KB of text data (about 150 words) are information equivalent to 1.2 MB of voice-audio data. In this test, freeware is used as a plug-in component in Gateway to convert data from voice to text on the publisher's request. The subscriber connects to the JMS server directly, and converts received text data to voice. The measured time is from the publisher to the subscriber (endpoint to endpoint) including voice-text and text-voice conversion time. Publishing more messages causes an impact on performance from the overhead of the conversion process and transmission. However, it is clear that transforming voice data to text results in a dramatic reduction of data size, which provides high performance.

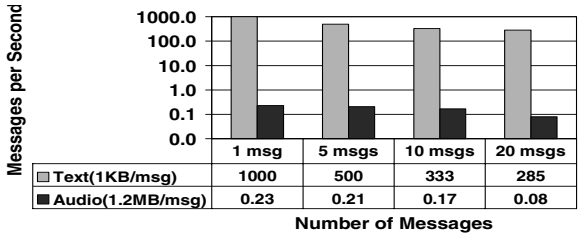

Figure 11: Audio/Text Semantic Transcoding

Note: The PCs used for the above testing had X86 (Pentium III) 256-392MB RAM 600MHz-800MHz with Windows2000/Professonal or Linux 6.2Redhat.

5. Summary and Discussion

Gateway and SmartCaching aim to solve emerging difficult design issues of a messaging system in mobile/wireless environments. Gateway can deploy different plug-in functions such as semantic transcoding, caching, and compres-

sion for message optimization. The message subscribe and the snapshot service of SmartCaching give better flexibility for the design of mobile applications and allow mobile-specific constraints to be dealt with. Some interesting plug-in components give significant performance improvements. Gateway provides an intelligent message hub for reliable and efficient transmission by integrating a Mobile-JMS Client, SmartCaching, and various plug-in components.

5.1 Related Work

Optimizing data over a wireless environment has been successful although technologies are tightly coupled with the applications or the servers based on a client-server model. Techniques for optimization include caching, protocol reduction, and adding an asynchronous model [8]. For example, IBM's WebExpress [5, 7] provides a web browser proxy between mobile clients and a web server to optimize HTTP data. IBM's WebSphere Transcoding Publisher [2] is a server-based software that dynamically translates web content and applications into markup languages [9]. However, none of these can be deployed in a dynamic distributed environment. Caching is also tied to applications in most cases. Java Temporary Cache (JCache) [13] has been proposed (but not yet implemented practically) by Oracle, providing a standard set of APIs and semantics that are the basis for most caching behavior [3] including N-tier support. Softwired's iBus/Mobile [11] extends JMS to mobile-tier and is designed as an extension of J2EE application servers such as BEA WebLogic [1]. In contrast, Gateway is a simple message hub that can reside on the device or anywhere between clients and servers. Combining Gateways into a powerful message-hub network, provides a flexible N-tier layout. This is a novel distributed system approach for messaging over a mobile-tier instead of through tight linkage with a server.

5.2 Future Work

Application-specific objects are instantiated by an application, register publicity and are then used by other applications as remotely accessed distributed objects. Topics for publish/subscribe, and the configuration of Gateways for filtering and transformation, fit well with this scheme. Java Naming and Directory Interface (JNDI) is a Java technology API for publishing, managing, and accessing public references to distribute functionality. Currently JNDI is not supported in Java Micro Edition and a standard API for this function over a mobile environment will be critical. This includes security aspects such as encryption, authentication, and access control on distributed objects. In SmartCaching, a synchronization mechanism will be needed to propagate the changes that it receives. It also needs persistent storage for cached data. Generic persistent storage over a distributed system, specific for mobile/wireless environments, would therefore be useful.

Acknowledgment. We thank Jon Crowcroft (University of Cambridge) for critical reading and constructive comments.

References

[1] BEA Systems. WebLogic 7.0 Java Message Service. http://www.bea.com/products/index.shtml/.

[2] K. H. Britton *et al.* Transcoding: Extending e-business to new environments. *IBM System Journal*, Vol.40(No.1), 2001.

[3] M. Butrico *et al.* Gold Rush: Mobile Transaction Middleware with Java-Object Replication. In *3rd Conference on Object-Oriented Technologies and Systems (COOTS)*, 1997.

[4] L. Chalamtac. *Wireless and Mobile Network Architecture*. Wiley, 2001.

[5] H. Chang *et al.* Web Browsing in a Wireless Environment: Disconnected and Asynchronous Operation in ARTour Web Express. *MOBICOM: Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 260–269, 1997.

[6] P.Th. Eugster *et al.* The Many Faces of Publish/Subscribe. *Technical Report TR-DSC-2001-04, Swiss Federal Institute of Technology*, January 2001.

[7] B. Housel and D. Lindquist. WebExpress: A System for Optimizing Web Browsing in a Wireless Environment. *Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking*, pages 108–116, 1996.

[8] J. Jing, A. Helal, and A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, Vol.31(No.2), 1999.

[9] C. Lau and A. Ryman. Developing XML Web services with WebSphere. *IBM System Journal*, Vol.41(No.2), 2002.

[10] K. Nagao. Semantic Transcoding: Making the World Wide Web More Understandable and Usable with External Annotations. In *Proceedings of International Conference on Advanced in Infrastructure for Electronic Business, Science, and Education on the Internet*, 2000.

[11] Softwired. iBus Messaging. http://www.softwired-inc.com/.

[12] Sun Microsystems. *Java Message Service (JMS) API Specification*. http://java.sun.com/products/jms/.

[13] Sun Microsystems. *JCache: Java Temporary Caching API*. http://www.jcl.org/jsr/detail/107.prt.

[14] Sun Microsystems. *RMI Profile Specification on Connected Device Configuration (CDC)*. http://java.sun.com/aboutjava/communityprocess/jsr/.

[15] Eiko Yoneki and Jean Bacon. Pronto: MobileGateway with Publish-Subscribe Paradigm over Wireless Network. Technical Report UCAM-CL-TR-559, Computer Laboratory, University of Cambridge, also to appear in ACM/IFIP/USENIX International Middleware Conference (Work in Progress), June 2003.