Technical Report

UCAM-CL-TR-559
ISSN 1476-2986

Number 559

Pronto: MobileGateway with publish-subscribe paradigm over wireless network

Eiko Yoneki, Jean Bacon

February 2003

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Series editor: Markus Kuhn

ISSN 1476-2986

Pronto: MobileGateway with Publish-Subscribe Paradigm over Wireless Network

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. This paper presents the design, implementation, and evaluation of **Pronto**, a middleware system for mobile applications with messaging as a basis. It provides a solution for mobile application specific problems such as resource constraints, network characteristics, and data optimization. **Pronto** consists of three main functions: **1- MobileJMS Client**, 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 with pluggable components, and **3-Serverless JMS** based on IP multicast.
The publish-subscribe paradigm is ideal for mobile applications, as mobile devices are commonly used for data collection under conditions of frequent disconnection and changing numbers of recipients. This paradigm provides greater flexibility due to the decoupling of publisher and subscriber. Adding a gateway as a message hub to transmit information in real-time or with store-and-forward messaging provides powerful optimization and data transformation. Caching is an essential function of the gateway, and **SmartCaching** is designed for generic caching in an N-tier architecture. Serverless JMS aims at a decentralized messaging model, which supports an ad-hoc network, as well as creating a high-speed messaging BUS. **Pronto** is an intelligent **MobileGateway**, providing a useful MOM intermediary between a server and mobile devices over a wireless network.

Keywords: Gateway for mobile and wireless systems, Message oriented middleware, Publish-Subscribe, JMS, SmartCaching, Ad-hoc Network

1 Introduction

Computing devices are becoming increasingly mobile at the client end, and they are based on new communication models. A large-scale distributed system must offer load sharing and load reduction for good performance. These two key issues are more complex for mobile/wireless-based applications and web-based services. In a mobile/wireless network environment, latency is high, bandwidth is low, and the connection can be interrupted at any time. Mobile devices have a small footprint and often use different transport mechanisms to connect to the network, and many devices are not programmable. The architecture of a distributed system at this level needs careful consideration of many factors, including client applications, middleware tiers, and network characteristics all the way up to the server. It is essential to provide the core function for such a system as semantics-based middleware.

This paper presents **Pronto**, a middleware for mobile applications. The basis of Pronto is a Message Oriented Middleware (MOM) based on Java Message Service (JMS) [33] in both centralized and decentralized forms. Pronto introduces a gateway for reliable transmission and efficiency between mobile applications and servers, taking advantage of plug-in components for caching, device-specific transport, compression, semantic transcoding, and message transformation, as well as supporting disconnected operation. Pronto provides a useful MOM layout from a server to mobile devices over a wireless network, and its performance is optimized by applying and comparing different techniques. To demonstrate the potential of Pronto, several applications were written. Below, the three main functions forming the basis of Pronto are described.

MobileJMS Client: Several communication mechanisms such as Remote Procedure Call (RPC) and Remote Method Invocation (RMI) have been used for sharing the workload in distributed systems. JMS is a recent Java technology, providing an Application Programming Interface (API) for inter-client communication among distributed applications. It provides a communication mechanism that differs from other mechanisms such as EJB (Enterprise Java Beans), Sun's JINI, CORBA (Common Object Request Broker Architecture), which define a higher level of logic for applications. JMS is a service-oriented API specification, prescribing messaging functionality in terms of interfaces. It offers publish-subscribe and point-to-point paradigms and expands previous messaging capabilities [1]. It is now evident that peer-to-peer or client-server communication models cannot cover all distributed inter-process communication: intercommunication is commonly achieved using directed links between tightly coupled senders and receivers, where the message destination must be known at the time of sending, which is difficult with changing destinations or varying numbers of recipients. By contrast, MOM encourages 'loose coupling' between message producers and message consumers with a high degree of anonymity, giving the advantage of the removal of static dependencies in a distributed environment. This permits a dynamic, reliable, and flexible system to be built. JMS provides the architecture for MOM and it is implemented in Pronto. The challenge is to accommodate JMS with mobile devices less powerful than workstations. The specifications and interfaces of JMS are complex, but not all functions are mandatory for a mobile/wireless environment. Thus, considering the mobile device peculiarities, one of the aims is to create a mobile-specific JMS client API. More widespread use of mobile devices will generate an increasing need for decoupling of distributed objects. MOM adds high scalability and easy integration into heterogeneous networks. This is described in more detail in Section 3-4.

Gateway and SmartCaching: Given the characteristics of mobile devices and wireless networks, more work is required for better performance. Some important points are specified below:

– Wireless networks become increasingly packet-oriented, and reducing the size of data for transmission is beneficial.
– Because of low bandwidth, high latency, and frequent disconnections, 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 should give advantages for efficient data flow.
– Many devices are non-programmable, and a gateway has to bridge messages between non-programmable devices and a server.

In Pronto, an intermediate gateway (called **Gateway**) is introduced as a 'traffic controller' to provide reliable transmission and efficiency between clients and a server through pluggable components. Plug-in components themselves are not discussed in this paper except for caching. SmartCaching is designed and used in Gateway (see Section 5-6 for details).

Serverless JMS: A goal of Serverless JMS is to adapt JMS into a decentralized model. This model will perform best over an ad-hoc network, and a high-speed transmission of a large number of messages, e.g. to distribute the workload of a server to several servers. Many underlying transmission media such as Ethernet provide support for multicast and broadcast at the hardware level. Implementing Serverless JMS using the IP multicast service [26] over such a network leads to a significant performance improvement (see Section 7 for details).

2 Background

This section provides a brief overview of the characteristics of mobile/wireless networks, mobile devices, and JMS.

2.1 Mobile/Wireless Networks

In a mobile/wireless network [6] a communication is not only connection-based, but also packet-oriented. With a packet-oriented bearer such as GPRS (General Packet Radio Service) or UMTS (Universal Mobile Telecommunications System), a device can send and receive information packets without dialling into a network service provider. With packet data, users typically pay for the time they communicate data and not for idle time. Packet-oriented bearers are better suited for the connectionless model of JMS, and reduction of the amount of data transmitted is crucial. The ad-hoc network, another feature of a mobile/wireless network, is a dynamically re-configurable wireless network without a fixed infrastructure that does not require the intervention of a centralized access point. Thus, the network devices are only part of the network during communication sessions. The message domain publish-subscribe in JMS can reside on an ad-hoc network, but not the implementation based on a centralized server model of JMS. Here Serverless JMS will play an important role.

2.2 Mobile Devices and Mobile Applications

Mobile devices have different characteristics from workstations and specific issues that need to be addressed are described below:

- Mobile devices have small ROM and RAM footprints as well as low usage of CPU cycles and battery power. A client library of a middleware should therefore have a small memory footprint.
- Applications on mobile devices frequently lose and regain network connectivity. Middleware should provide a direct or indirect function to insure ongoing communication or provide an interface to applications that allow the maintenance of communication during the disconnect operation.
- There are various bearers such as 2G, 2.5G, 3G, Bluetooth, and IEEE 802.11. Middleware needs to offer an interface of communication abstraction on top of various wireless bearers.
- There are different operating systems on mobile devices and, thus far, no one has managed to dominate the market. A multi-platform middleware should be implemented in the platform independent language Java and compatible with Java standards such as J2ME (Java 2 Micro Edition) [34] or Personal Java. Recent developments of J2ME help this aspect, and it supports the following two ranges of mobile devices:

  Pocket PC devices: for example, Compaq iPAQ or Sharp Zaurus PDA containing high-speed processors with a good amount of memory. J2ME Connected Device Configuration (CDC) runs on this range of devices. CDC is a full feature Java2 VM building block for the next generation of consumer electronic and embedded devices. RMI [36] is supported.
  Small footprint J2ME CLDC capable devices: for example, Nokia 6310i, Motorola i85, or Personal Digital assistant (PDA) such as the Palm. Sun's K virtual machine (KVM) is designed for these devices. Combined with a set of device-specific Java API, such as Mobile Information Device Profile (MIDP) and Connected Limited Device Configuration (CLDC), J2ME provides a complete runtime environment for small resource-constrained devices.

- Although an applet or a web service is not a mobile device, it has similar characteristics.

2.3 Java Message Service (JMS)

JMS defines a common set of interfaces and associated semantics, providing a common way for Java programs to create, send, receive, and read messages. The initial version 1.0 specification was released by Sun Microsystems in 1998 [33, 21]. The messaging service provides support for passing messages between distributed applications in a reliable, asynchronous, loosely coupled, and language-and-platform independent manner. The common building block of a messaging service is the message. The message consists of events, requests and replies that are created by and delivered to clients. In JMS, messages come in several types such as BytesMessage, MapMessage, ObjectMessage, TextMessage, and StreamMessage. Not all message types are essential in the mobile-tier. Messaging services such as persistent delivery, durable subscription, the time-to-live option on a message, and transactions display the range of delivery methods. Asynchronous messaging is a key integration point in a mobile/wireless environment. JMS defines two messaging paradigms, publish-subscribe and a point-to-point, the latter being less suited for mobile-tier. JMS does not define standard address syntax, using instead the *Destination* object that encapsulates the address [10]. In the publish-subscribe paradigm, the *Destination* is called Topic. Producers send messages to the *Destination*, which in turn delivers messages to consumers. Messages are sent to the *Destination* rather than specific processors or ports. Communication is typically one-to-many and asynchronous. The publish-subscribe paradigm supports the development of location-independent applications that can be moved from one machine to another without affecting their peer applications. JMS works well in an environment where network connections sometimes break, and the available bandwidth can vary within a short time. MOM's characteristics (intuitive programming model, latency hiding, guaranteed delivery, store-and-forward) are highly appealing for mobile applications. The JMS server side is typically implemented by Java application servers, which provide a host of Java technology-based services, as well as by MOM, which operates either alone or in concert with other distributed computing software.

3 System Overview

As discussed above, the publish-subscribe communication paradigm is well suited for a mobile environment, as mobile devices are used for data collection under conditions of frequent disconnection and changing numbers of recipients. Adding a gateway as a message hub to transmit information in real-time or with store-and-forward messaging will give more powerful optimization of data reduction and transformation including a caching function. Pronto is designed as a middleware, forming a collection of generic distributed services that are application-independent. The implementation of Pronto is 100% in Java and it consists of the following 4 main packages with different Java platform profiles such as J2SE, J2ME with+CDC and J2ME+CLDC:

- **MobileJMS Client:** A lightweight JMS client library.
- **Gateway:** A bridge for mobile clients and a framework for optimization functions, implementing two modes: 'Local' and 'Remote'.
- **SmartCaching:** Generic caching function for N-tier is used in Gateway.
- **Serverless JMS:** MobileJMS client in the decentralized model.

Currently a simple JMS server was implemented for Pronto to support MobileJMS Client, which is out of scope of this paper.

3.1 Distributed Systems with Pronto

Fig. 1 shows an overview of a distributed system with Pronto. Four different deployment possibilities are illustrated:

A - Application with MobileJMS Client: An application in a mobile device uses a MobileJMS Client API. It communicates directly with the JMS server.
B - Application with MobileJMS Client and LocalGateway: An application in a mobile device uses MobileJMS Client API. LocalGateway can run as a separated thread from the application or within the application and performs caching and transcoding through plugged-in components.
C - Application with MobileJMS Client and RemoteGateway: An application in a mobile device uses a MobileJMS Client API. RemoteGateway is

running as a separate process. Currently RMI-based transport between a RemoteGateway and MobileJMS Client is implemented.
D - Non-Programmable Devices with RemoteGateway: Non-programmable devices require RemoteGateway to perform proper transportation and message transformation for the target device. RemoteGateway represents every subscriber and publisher for the non-programmable device. A non-programmable device can be any entity as far as it implements the *Transport* interface defined by Gateway.

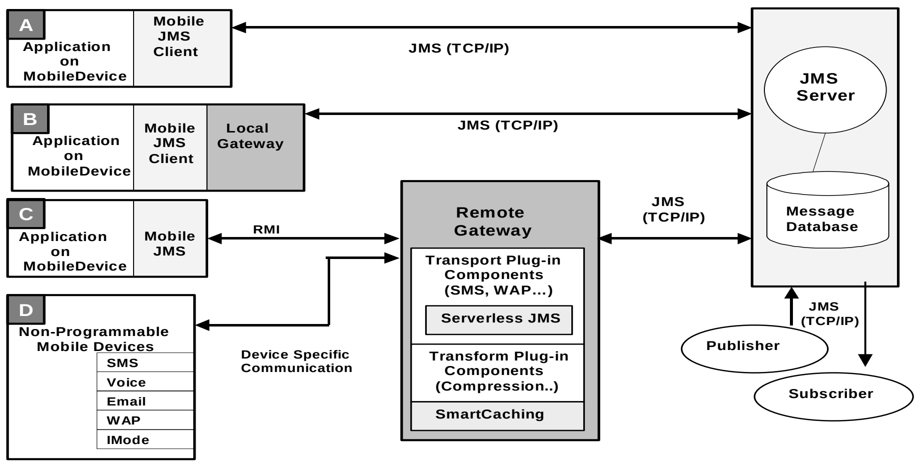

A
Application on MobileDevice
Mobile JMS Client
JMS (TCP/IP)
B
Application on MobileDevice
Mobile JMS Client
Local Gateway
JMS (TCP/IP)
C
Application on MobileDevice
Mobile JMS
RMI
D
Non-Programmable Mobile Devices
SMS
Voice
Email
WAP
IMode
Device Specific Communication
Remote Gateway
Transport Plug-in Components (SMS, WAP...)
Serverless JMS
Transform Plug-in Components (Compression..)
SmartCaching
JMS (TCP/IP)
JMS Server
Message Database
JMS (TCP/IP)
Publisher
Subscriber

Fig. 1. System Overview of Distributed System with **Pronto**

3.2 Disconnect Operations

Most work dealing with the disconnectedness of computing devices revolves around data replication and synchronization. Data replication involves copying data from a central repository to a mobile device. Synchronization is a similar concept but also allows for changes made in either copy of the data to be propagated to the other end. In Pronto, the following approaches are designed for disconnected operation:

- **Durable subscription in JMS:** 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 be durable as an option by registering a durable subscription with a unique identity.
- **Gateway Cache:** Gateway maintains the cache even if applications are inactive. Applications can use the Gateway cache after regaining the connection. The applications can use the 'pull', 'subscribe', and 'snapshot' operations of SmartCaching at appropriate occasions. For example, an application spawns a background activity that synchronizes the on-device messages when connected. The application can choose to use either LocalGateway cache or RemoteGateway cache.

3.3 Serverless Model

A JMS scheme is provided as a decentralized model: Serverless JMS. A publisher acts as a temporary server and keeps a subscription list. Serverless JMS can be embedded in Gateway as a plug-in component. Fig. 2 shows the message flow.

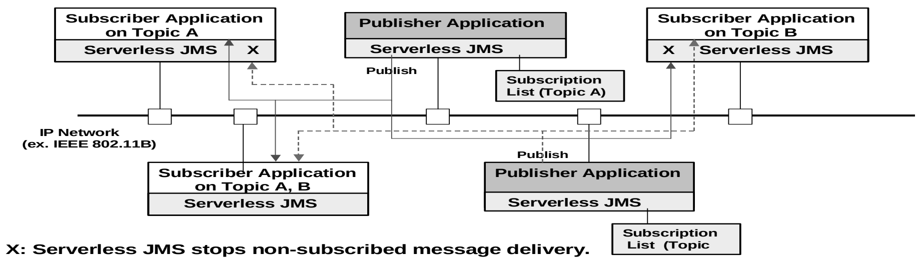

Subscriber Application on Topic A
Serverless JMS
X
Publisher Application
Serverless JMS
Publish
Subscription List (Topic A)
Subscriber Application on Topic B
X
Serverless JMS
IP Network (ex. IEEE 802.11B)
Publish
Subscriber Application on Topic A, B
Serverless JMS
Publisher Application
Serverless JMS
Subscription List (Topic
X: Serverless JMS stops non-subscribed message delivery.

Fig. 2. Serverless JMS over IEEE 802.11 Network

3.4 Gateway Cascade

Gateway can be used to distribute JMS messages to the target Gateways where they are sent to the devices. JMS BUS is a Serverless JMS over a high-speed bus. A high-speed bus can be LAN-based or WAN-based as far as the routers allow IP multicast. A deployment example is shown in Fig. 3.

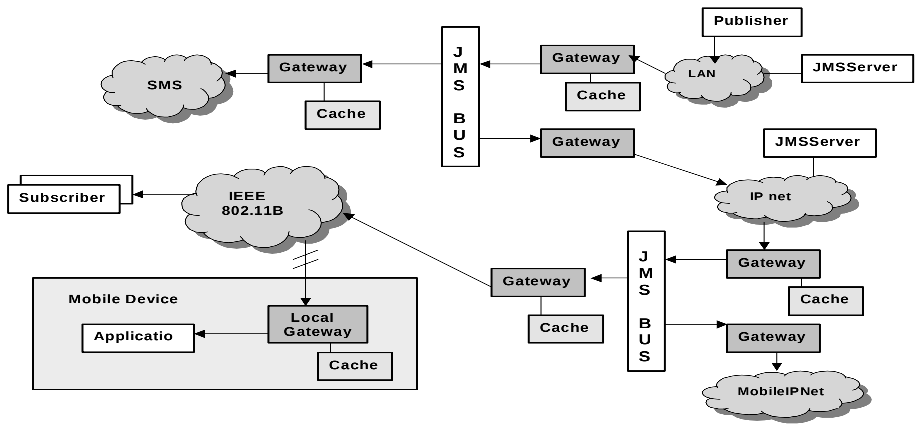


Fig. 3. Gateway Cascade

4 MobileJMS Client

MobileJMS Client is designed by following the JMS API model. In order to specialize JMS to a mobile environment the points below are considered:

Connection: A connection represents an open connection to the JMS server. JMS does not define any specific transport mechanism. In Pronto, HTTP via TCP/IP is implemented, which allows the applets using MobileJMS Client to

connect through firewalls to the JMS server. Most JMS servers provide HTTP connection, and J2ME supports HTTP.
Session: Connection creates Sessions. A session is a single-threaded context that handles message-passing operations. A JMS server has a session pool and can execute separate messages concurrently, thus improving performance. If a client code needs to receive asynchronous messages concurrently, the client can use multiple sessions to partition client operations, i.e., one client thread can drive one session while another client thread can drive another. Each session object serializes execution of message listeners. Thus, message listeners can share session-related resources. In order to avoid the complex threaded model, connections and sessions share one thread to receive a message in Pronto.
Message: A message is a lightweight object consisting of a header and body. The header contains identification and routing information and is optimized to be as small as possible. The body contains application data. Essential message types such as TextMessage, BytesMessage, and ObjectMessage from the five message types defined in JMS are implemented.
Thread-Safety: In general, the JMS specification requires a thread-safe implementation for all objects, but only *Destination*, *Connection*, and *ConnectionFactory* objects support concurrent use in Pronto.
Durable Subscription: A durable subscriber registers a durable subscription with a unique identity. A subsequent subscriber with the same identity resumes the subscription in the state left by the previous subscriber. If there is no active subscriber for a durable subscription, JMS retains the subscription's messages until they are received by the subscription or until they expire.
Message Flow: For durable/persistent messages, each message is stored by the JMS server before delivery to the consumer and is removed after delivery. This has a huge impact on performance. The message has an expiration time from the time-to-live beyond the time of publication. For non-durable/non-persistent messages, the time for delivery to the *Destination* depends on message numbers and *Destination* sizes. Redelivery delay time defines when to redeliver a message after a failure. With shorter times, the frequency of redelivery is high, thus increasing network traffic. Pronto follows the design of the message flow as described, and it is important to set a sensible time interval to improve performance over a mobile/wireless environment.
Message Selector (Content Based Subscription): Topics can be structured into hierarchies, and subscriptions can be a part of the hierarchy. This provides content-based messaging and greater flexibility for applications as there is less coupling between producers and consumers. Content-based addressing is more consumer-oriented, whereas subject-based addressing is producer-oriented. A topic hierarchy is not part of the JMS specification, but it can be effective in a mobile/wireless environment to control traffic. The current Pronto provides 'Message Selector' for content-based subscription. Message Selector is a filter for a topic defined by the consumer. The JMS server evaluates Message Selector and does not deliver non-qualifying messages. In Pronto, this filter is implemented within XML based TextMessage. A message selector is a string, whose syntax is based on a subset of SQL92 conditional expression syntax. In the example of Fig. 4, only the second message published will be delivered to the subscriber.

```
Publisher:
   . . . . .
   TextMessage textmessage = session.createTextMessage
   (‘<?xml version=\’1.0\’ encoding=\’UTF-8\’ ?><List><Millionaire Name=\’Bahler Income=500\’/></List>’);
   publisher.publish(tm, Message.PERSISTENT);
   textmessage = session.createTextMessage
   (‘<?xml version=\’1.0\’ encoding=\’UTF-8\’ ?><List><Millionaire Name=\’Gates Income=10000\’/></List>’);
   publisher.publish(tm, Message.PERSISTENT);
Subscriber:
   . . . . .
   connection.start();
   subscriber = session.createXMLSubscriber(topic, ‘Millionaire.Income >= 5000 ’);
   subscriber.setMessageListener(new Listener(subscriber));
```

Fig. 4. Message Selector Use

5 Gateway

In an N-tier architecture, proximity to the data source requires a finer granularity of filtered subscription to identify the data, while increasing distance to the source makes the data more localized. All data requests converge onto a data source that has to identify the needs of requesters. Close to the periphery, each requestor already knows the needs, be it through object identity or predicate definition. Thus, the counter-intuitive principle is to have a fine-grain subscription close to the data source and a coarse-grain subscription further away from the source. Gateway is a message hub and distributes messages into several Gateways and applications, and messages commonly contain requests and responses to/from the data source.

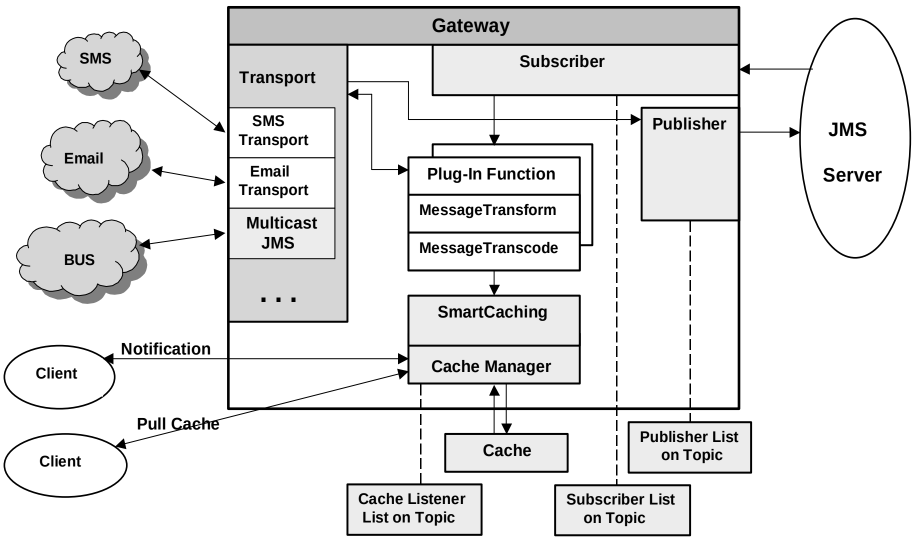


Fig. 5. Overview of Gateway Operation

An overview of the operation of Gateway is shown in Fig. 5. 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. For non-programmable devices, Gateway defines a *Transport* interface to communicate with non-programmable devices. Gateway technology is based on a ’store-and-forward’ communication model, transforming messages above the transport level. This offers load sharing and load reduction for good performance. All plug-in components are defined in the XML configuration file. The configuration information is shared between Gateway and the client applications. Note that the configuration information has to be managed (e.g., by JNDI) to be accessible by both parties. Any specific configuration utility is not yet implemented in Pronto.

Plug-In Components: To gain performance for a distributed system, 'server performance', 'network and middleware latency', and 'presentation and transformation complexity' have to be considered. Caching, compression, and semantic transcoding are good candidates to reduce data size and network traffic and suitable for the plug-in functions. Security (encrypting/decrypting data) functions can also be plugged into Gateway.
Semantic Transcoding: Semantic transcoding offers more than simple image data transcoding. 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 in the mobile environment is important. In semantic transcoding the data are linked to an annotation [22]. Annotations can be corresponding text for a video clip, a summary of a document, greyscale/downsized/low-resolution image data, or a linguistic annotation of the content for voice synthesis.
Local and Remote Gateways: Gateway itself is defined as an interface, and two implementations are available. LocalGateway resides in a mobile device aside the MobileJMS client and can be instantiated as a separate thread. RemoteGateway resides between the JMS server and the client and is currently implemented using RMI. RMI is part of the J2ME/CDC package. Mobile devices can take advantage to use either LocalGateway or RemoteGateway depending on the applications.
Non-Programmable Transport: *Transport* is an interface to manage the non-programmable devices. The registration of a *Transport* to Gateway activates a subscription to a JMS server on the specified topic. Messages that are delivered to Gateway will be forwarded to the *Transport*, which looks up the device list and session list, and sends messages accordingly. Messages published via the *Transport* are forwarded to JMS server. Fig. 6 shows the control flow of the *Transport* interface.

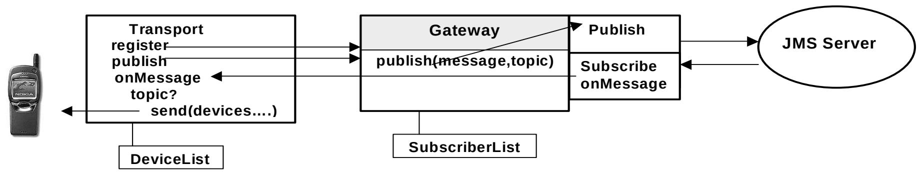


Fig. 6. Non-Programmable Devices and RemoteGateway

6 SmartCaching

Caching is an essential function in Pronto, leading to performance improvements by reducing network traffic and improved latency. The cached data can be raw or processed and stored for reuse without the need for revisiting the server and passing the data through the chain of reformatting and representation. An extensive cache function supports multi-tiered applications across platforms and devices. In Pronto, SmartCaching is designed as a separate package to make it generic and independently usable. In Pronto, only basic functions are currently implemented, while persistent caching, cache validation, synchronization, and coherency management are beyond the scope of this study. Key functions are:

– **Pull cache:** Pull the whole stored cache.
– **Subscribe Cache:** Receive event notification when the cache is updated.
– **Snapshot:** Keeps the last image of the cache

SmartCaching is read-only and decoupled from the data source, and the cache can be active or up-to-date. Thus the application does not need to request to pull the data that have already been requested from the data source again. Applications now become event-driven and active. This simple change has a major impact on performance and scalability 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 that creates *Cache* objects and manages requests and responses to the requesters. *Cache* is an object that contains a key and actual caching objects that are kept as a linked list. The *Cache* object contains the expiration date, and expired objects will be removed by the *CacheManager*. 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*. In Pronto, Gateway embeds SmartCaching to store JMS messages.

Pull Cache: An application requests a cache in synchronous mode (Fig. 7).

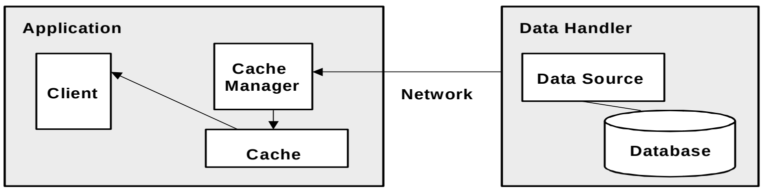


Fig. 7. SmartCaching: Pull Cache

Subscribe Cache: An application requests a cache update notification to a cache handler, who notifies the application after the cache is updated(Fig. 8).
Snapshot: When data are delivered to applications in a time series, clients should be able to reconstruct the latest view of the information of interest. This can be achieved by requesting a re-broadcast from the data source or by retaining the last image in a shared cache. The second option corresponds to 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, Snapshot rule is applied and persists in a cache. The rule for the Snapshot 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 Snapshot arrival, to apply the data to

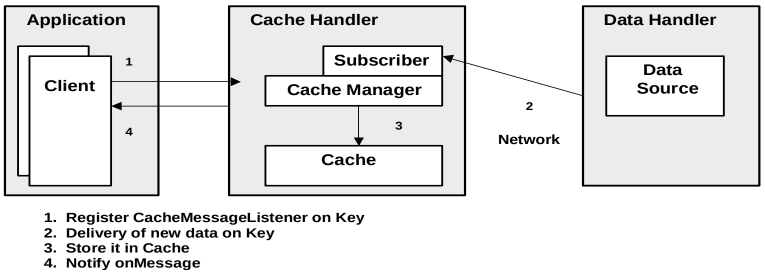


Fig. 8. SmartCaching: Subscribe Cache

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 information 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 messages. 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 Fig. 9.

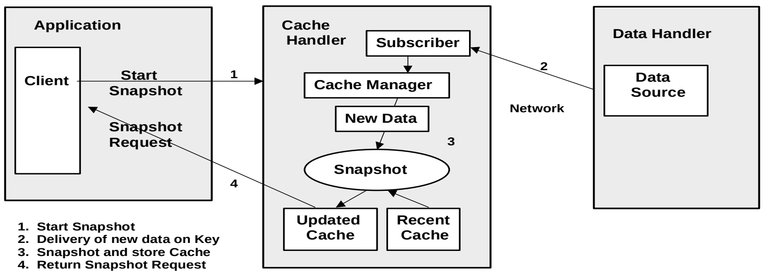
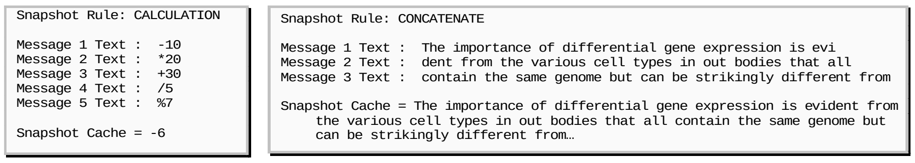


Fig. 9. SmartCaching: Snapshot Data Flow

Fig. 10 shows examples of Snapshot rules. In the first example, a message contains a delta value from the Snapshot base, and the rule is simply to carry out an arithmetic operation (in this case, the base is 0). In the second example, a message is added to the tail of the previous one.


```
Snapshot Rule: CALCULATION

Message 1 Text :  -10
Message 2 Text :  *20
Message 3 Text :  +30
Message 4 Text :  /5
Message 5 Text :  %7

Snapshot Cache = -6
```

```
Snapshot Rule: CONCATENATE

Message 1 Text :  The importance of differential gene expression is evi
Message 2 Text :  dent from the various cell types in out bodies that all
Message 3 Text :  contain the same genome but can be strikingly different from

Snapshot Cache = The importance of differential gene expression is evident from
     the various cell types in out bodies that all contain the same genome but
     can be strikingly different from…
```

Fig. 10. Examples of Snapshot Rules

7 Serverless JMS

Serverless JMS is a serverless version of MobileJMS Client. It delivers messages over IP multicast with high performance and reliable message delivery.

The aim is to put the JMS scheme in a decentralized model, using IP multicast as transport mechanism. Considering JMS, IP multicast allows you to transmit messages from one publisher to many subscribers efficiently and without a redundant network traffic. The number of subscribers can increase without an impact on network traffic. Some JMS features (e.g., the point-to-point paradigm and durable subscription) were omitted given the nature of the network model and IP multicast protocol.

Multicast Group: Groups of machines representing a multicast group are identified by an IP multicast address. Each address can be considered as a 'channel' to identify groups of hosts interested in receiving the same content. Two channels are used in Serverless JMS. The ManagementChannel is used for administration purposes, while the MessageChannel is used for message transmission. As an option, MessageChannel can be defined on each topic.
Reliable Protocol: The basic service provided by IP multicast is an unreliable datagram multicast service, and there is no guarantee that a given packet has reached all intended recipients of a multicast group. Serverless JMS implements both reliable and unreliable multicast transports. The reliable version uses a negative acknowledgement-based reliable protocol. The transparent fragmentation and re-assembly of messages that exceed a UDP datagram size is implemented. This provides the highest possible delivery guarantee in a multicast environment [39, 27, 29].
Flow Control: The speed of the modern LAN transmission is high, and the packet loss will be rare with a good network quality. However, due to the high speed, the buffer is overwritten and messages will be discarded if the network buffer is not large enough and the subscriber cannot keep up with the speed of incoming data. This corresponds to packets being lost during the transmission. The window based flow control between publishers and subscribers is implemented.
Subscription Registration: Two subscription modes are defined: the administrated and non-administrated modes. In the non-administrated mode, publishers publish messages independently of the existence of subscribers.
Auto Discovery: An auto discovery function is designed. A publisher runs an independent thread for auto discovery, which sends management data that require an echo from subscribers via ManagementChannel and maintains the subscription list. Auto discovery repeats this at defined intervals.

8 Experiments and Results

MobileJMS Client allows for seamless use of the same API with J2SE, J2ME/CDC, J2ME/CLDC, and Serverless JMS packages. Examples below demonstrate the capability of Pronto. Considering that Pronto provides a middleware, a full evaluation would require a large-scale integration test, including 3G networks with hundreds of mobile devices connecting to an application server with the database, which is out of the scope of this paper.

8.1 Demo Applications

Time Series of Video Data Publishing over 802.11B Network: A video camera takes 15 seconds of video every 30 seconds, and data are published under the topic 'Demo'. All subscribers receive the published video data. An iPAQ user is moving, leading to occasional disconnections. LocalGateway running in iPAQ is set to durable subscription, and all published data are processed by the application. LocalGateway's caching provides the entire process without significant delay (Fig. 11).

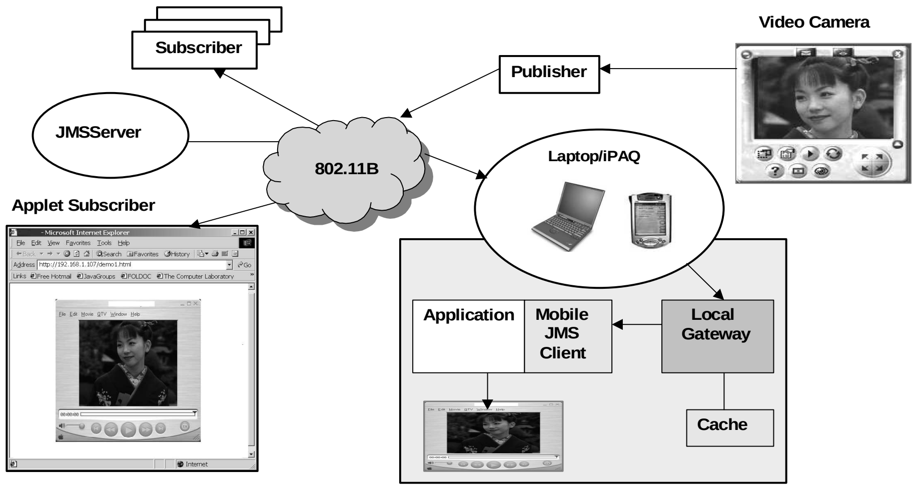
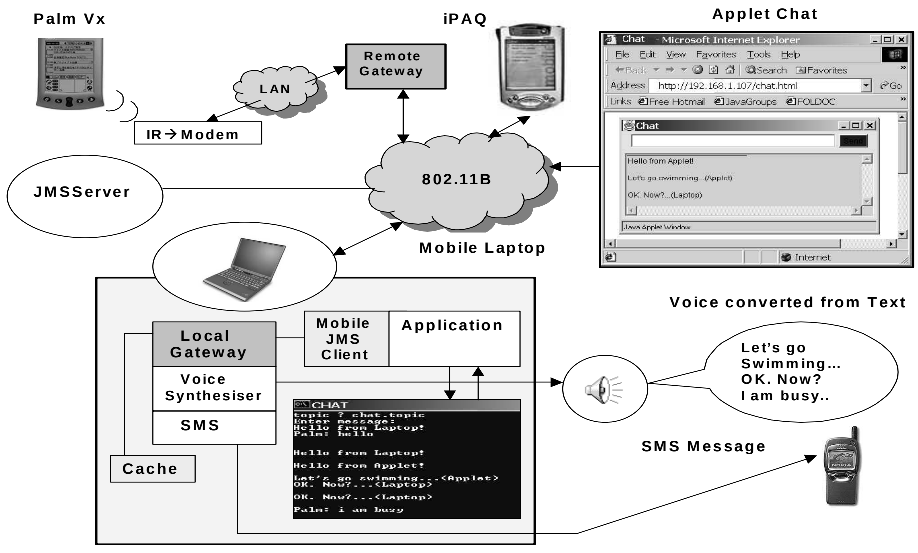


Fig. 11. Demo: Time Series of Video Data Publishing over 802.11B Network

Voice/Text/SMS/Applet/Palm Chat: Chat among an applet, an application on a laptop, Palm pilot, iPAQ, and SMS phone. All the clients subscribe to the topic 'Chat'. A Palm can connect via IR/Modem into LAN. A Gateway processes two plug-in components 'Voice Synthesizer' and 'SMS' (Fig. 12).



```
Hello from Applet!

Let's go swimming...(Applet)

OK. Now?...(Laptop)
```

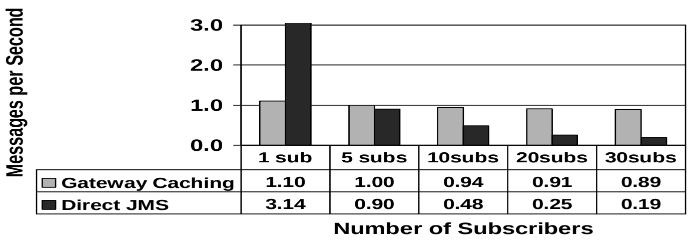
```
topic ? chat.topic
Enter message:
Hello from Laptop!
Palm: hello

Hello from Laptop!

Hello from Applet!

Let's go swimming...<Applet>
OK. Now?...<Laptop>

OK. Now?...<Laptop>

Palm: i am busy
```

Fig. 12. Demo: Chat among Voice/Text/SMS/Applet/iPAQ/Palm

8.2 Benchmark Test over 802.11B Network

A few samples from the benchmark test are given below. The PCs used for testing had X86 (Pentium III) 256MB-392MB RAM 600MHz-800MHz with Windows2000 Professional or Linux 6.2Redhat, and Sun Java2 SDK1.3.0.

Caching: This test focuses the performance of SmartCaching in RemoteGateway. A publisher publishes 50 KB x 20 BytesMessages. RemoteGateway subscribes to these messages and caches them. A subscriber listens to the cache update notification from RemoteGateway. The cache update notification gives better performance with more than one subscriber, and an increase in the number of subscribers does not have significant impact on performance (Fig. 13).

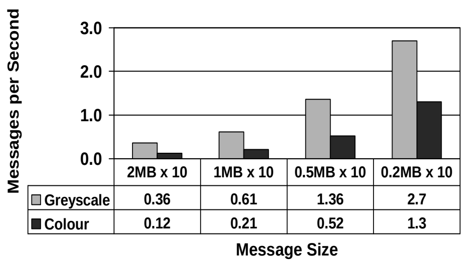


	1 sub	5 subs	10subs	20subs	30subs
▨ Gateway Caching	1.10	1.00	0.94	0.91	0.89
■ Direct JMS	3.14	0.90	0.48	0.25	0.19

Fig. 13. Performance improvement by RemoteGateway Caching

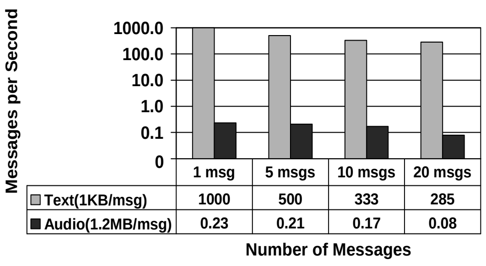
Image Semantic Transcoding: Transcoding colour image to greyscale image shows a significant performance improvement. In the test shown in Fig. 14(a), the conversion from colour to greyscale decreases an about 35% in data size.

Text/Audio Semantic Transcoding: This test focuses on a performance improvement by the semantic transcoding. 1KB of text data (about 150 words) is information equivalent to 1.2 MB of audio data. In this test, a freeware voice synthesis is used as a plug-in component in Gateway. The subscriber converted audio data to an artificial voice and the measured time includes text-voice conversion time. The size of the text message is negligible compared to the voice message.(Fig. 14(b)).



	2MB x 10	1MB x 10	0.5MB x 10	0.2MB x 10
▨ Greyscale	0.36	0.61	1.36	2.7
■ Colour	0.12	0.21	0.52	1.3

(a)Image Semantic Transcoding

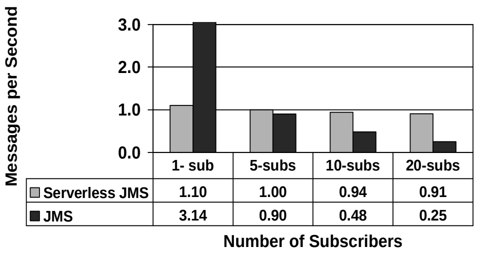


	1 msg	5 msgs	10 msgs	20 msgs
▨ Text(1KB/msg)	1000	500	333	285
■ Audio(1.2MB/msg)	0.23	0.21	0.17	0.08

(b)Audio-Text Semantic Transcoding

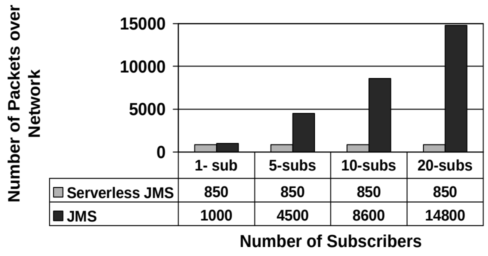
Fig. 14. Performance improvement by Gateway Plug-in Components

Serverless JMS: This test measures the capability of Serverless JMS. 250KB x 20 BytesMessages are to be published using Reliable option. No message retransmission occurred in this test.



	1- sub	5-subs	10-subs	20-subs
▨ Serverless JMS	1.10	1.00	0.94	0.91
■ JMS	3.14	0.90	0.48	0.25

(a)Throughput



	1- sub	5-subs	10-subs	20-subs
▨ Serverless JMS	850	850	850	850
■ JMS	1000	4500	8600	14800

(b)Network Traffic

Fig. 15. Performance comparizon between JMS and Serverless JMS

The results show that (Fig. 15):

– The number of subscribers does not have an impact on the performance in Serverless JMS, whereas regular JMS delivery shows an impact proportional to the number of subscribers.
– The number of packets over the network stays the same with increasing numbers of subscribers.

8.3 Scalability

JMS itself does not define 'load balancing', but the use of a clustering server is common. Clustering in the same geographical location is acceptable, but connectivity between different locations would need additional functionality. A Serverless JMS using a high-speed bus to transfer data between two different locations will be useful. Serverless JMS can function within the internet by establishing TCP bridges. Gateway Cascade (see Fig. 10) demonstrates the capability of Pronto to distribute from enterprise applications to a mobile environment. JMS defines attributes to improve performance, including concurrent message processing, different message types, and two types of message flow models. Deploying these options to integrate the system should in theory give good scalability.
RPC versus MOM: An N-tier system is a set of connected applications. Each tier has a dedicated thread that handles the requests from the previous tier and a process to forward the requests to the next tier. In MOM-based applications, several simultaneous requests can be sent to the target tiers using the same thread. An RPC-based application only submits a single request at a time to the next tier for each thread. A MOM-based application does not have to wait for responses, because requests are in the form of asynchronous messages. Since the maximum number of threads per application is constant, MOM-based applications are more scalable and faster.

9 Related Work

Since the initial JMS specification was released in 1998 [33], the existing MOM software have been rapidly integrated under the JMS API. Examples are IBM's

MQSeries [15], Microsoft Message Queue (MSMQ), TIBCO's TIB/Rendezvous [38], HP's Message Service [14], Softwired's iBus [32], and BEA's WebLogic [2]. However, Softwired's iBus/Mobile [20] is almost the only one to extend JMS to mobile-tier. iBus/Mobile is designed as an extension of J2EE application servers such as BEA WebLogic Server and JMS messaging products. It includes a messaging middleware client library compatible with the JMS standard as well as a middleware gateway used to connect mobile applications to J2EE application servers or other systems using JMS. It supports mobile communication specific protocols such as GPRS, UMTS, and CDPD. In Pronto, a gateway is a message hub that can reside in the device or anywhere in between. Pronto provides a flexible N-tier layout, deploying gateways instead of a tight linkage with a server. A gateway in Pronto offers more than a transport protocol as described in the above chapters. IBM's MQSeries Everyplace belongs to the MQSeries family of business quality messaging products. It is designed to satisfy the messaging needs of lightweight devices. There is no standard messaging API for the mobile environment [16].

The original iBus before the time of JMS used heavily multicast. Currently several JMS products support multicast transport such as TIB/Rendezvous. However, JMS has not been tried on mobile ad-hoc networks. Much research currently focuses on general datagram routing in both unicast and multicast routing [4, 25, 31, 18, 24], but no definite solution to provide JMS semantics using these protocols has been provided. Pronto uses multicast for JMS on ad-hoc networks. An example of a drawback of using multicast is the drastic performance reduction with redundant rebroadcasts [23]. For reliable protocol over IP multicast, various protocols such as SRM [11], RMTP [19], TRAM [9], and RMDP [30] are proposed and implemented. Pragmatic General Multicast (PGM) [28, 12] is a reliable multicast support protocol for applications that require ordered or unordered duplicate-free multicast data delivery from multicast sources to multiple receivers. For publish-subscribe messaging systems, PGM provides a building block for the messaging system itself, allowing higher performance and scalability for messages that need to go to many destinations. This is the most promising approach [37]. However, the PGM header is not yet supported by any Java package. For now a reliable protocol based on negative acknowledgement (NACK) is designed and implemented in Java in Pronto.

Optimizing data over a wireless environment has been successful. Most technologies are tightly coupled with the applications or the servers, based on client-server model. Techniques for optimization include caching, protocol reduction, header reduction, and adding an asynchronous model [7, 16]. For example, IBM's WebExpress [13, 8] provides a web browser proxy between mobile clients and a web server to optimize HTTP data. IBM's WebSphere Transcoding Publisher [3, 17] is a server-based software that dynamically translates web content and applications into multiple markup languages and optimizes it for delivery to mobile devices. Caching is also tied to applications in most cases. 'Java Temporary Cache' (JCache) [35] has been proposed by Oracle and provides a standard set of APIs and semantics that are the basis for most caching behavior [5] including N-tier support. Pronto provides an approach to integrate technologies to a compact semantics-based middleware in support of mobile/wireless environment specific issues.

10 Conclusion and Future Work

This paper points out various design issues of a messaging system for a mobile environment. Pronto is an effort to extend the messaging over a wireless network and solve the problems arising. Pronto provides broad functionality as described below.

MobileJMS Client is functional in resource constrained mobile devices that can cooperate with different wireless networks. The messaging paradigm fits well and the applications in mobile devices can publish-subscribe different types of messages and handle disconnected operation using durable subscriptions. Gateway can deploy different plug-in functions such as semantic transcoding, caching, and compression for message optimization. The message update notification and the snapshot service of SmartCaching give better flexibility for the design of mobile applications and allow to deal with mobile-specific constraints. Some interesting plug-in components give significant performance improvements. Serverless JMS can give additional functionality and power to JMS, the number of subscribers does not affect performance, and it seems an attractive model for an ad-hoc network environment. The benchmark test produced the expected results at this level. Note that JMS is more complex than discussed here. JMS would need support for administration, security, error handling and recovery, optimization, distributed transactions, and message ordering. Pronto could be extended in several directions described below, and none of them is defined in the JMS standard:

Network Lookup: It is important to have a standard API for registering and accessing distributed functionality. Application-specific objects were instantiated by an application, registered publicity and then used by other applications as distributed objects that have to be accessed remotely. Topics to publish and subscribe are good candidates to use this scheme. Java Naming and Directory Service (JNDI) is a Java technology API for publishing, managing, and accessing public references to distribute functionality. Currently JNDI is not supported in J2ME and a standard API for this function over a mobile environment will be critical.
Persistent storage: A JMS server will need persistent storage for the message queue, and the JMS client needs persistent storage for transaction processing. A generic persistent storage over a distributed system specific for mobile/wireless environments would therefore be beneficial.
Security: JMS API lacks security and encryption. There would be several aspects to consider, including message encryption, authentication, access control on distributed objects (*ConnectionFactory*, *Destination* etc.), and tunnelling messages across HTTP. Gateway can play an important role in security aspects.
Cache synchronization: The current SmartCaching is read-only, and a synchronization mechanism will be needed to propagate the changes that it receives.

In conclusion, Pronto shows how to extend the publish-subscribe paradigm over the mobile/wireless network in both centralized and decentralized forms. It provides an intelligent mobile gateway for reliable and efficient transmission by integrating MobileJMS Client, Serverless JMS, and various plug-in components. Pronto is therefore a powerful semantic-based middleware for messaging in mobile/wireless environment.

Acknowledgments. I would like to thank Jon Crowcroft (University of Cambridge) for critical reading and constructive comments.

References

1. Y. Aridor and M. Oshima. Infrastructure for Mobile Agents: Requirements and Design. In *Proceedings of the Second International Workshop on Mobile Agents*, pages 38–49, September 1998.
2. BEA. WebLogic 6.0 JMS. http://e-docs.bea.com/wls/docs60/jms/.
3. K. H. Britton, R. Case, A. Citron, R. Floyd, Y. Li, C. Seekamp, B. Topol, and K. Tracey. Transcoding: Extending e-business to new environments. *IBM System Journal*, Vol.40(No.1), 2001.
4. J. Broch, D. Johnson, and D. Maltz. *The Dynamic source Royting Protocol for Mobile Ad-Hoc Networks*, June 1998. draft-ietf-manet-dsr-02.txt (work in progress).
5. M. Butrico, H. Chang, A. Cocchi, N. Cohen, D. Shea, and S. Smith. Gold Rush: Mobile Transaction Middleware with Java-Object Replication. In *3rd Conference on Object-Oriented Technologies and Systems (COOTS)*, 1997.
6. L. Chalamtac. *Wireless and Mobile Network Architecture*. Wiley, 2001.
7. S. Chandra, C. Ellis, and A. Vahdat. Differentiated Multimedia Web Services Using Quality Aware Transcoding. *INFOCOM - Nineteenth Annual Joint Conference Of The IEEE Computer And Communications Societies*, 2000.
8. H. Chang, C. Tait, N. Cohen, M. Shapiro, S. Mastrianni, R. Floyd, B. Housel, and D. Lindquist. 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.
9. D. Chiu, S.Hurst, M. Kadansky, and J. Wesley. TRAM: Tree-based Reliable Multicast Protocol. *Sun Microsystems Technical Report TR-98-66*, 1998.
10. P.Th. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *Technical Report TR-DSC-2001-04, Swiss Federal Institute of Technology*, January 2001.
11. S. Floyd, V. Jacobson, and CG Liu. A Reliable Multicast Framework for Lightweight Session and Application Framing. *ACM SIGGOMM Computer Communications Review*, 1995.
12. J. Gemmell, T. Montgomery, T. Speakman, N. Bhaskar, and J. Crowcroft. The PGM Reliable Multicast Protocol. *IEEE Network special issue on Multicast:AnEnabllingTechnology*, 2003.
13. B. Housel and D. Lindquist. WebExpress: A System for Optimizing Web Browsing in a Wireless Environment. *MOBICOM: Proceedings of the 2nd Annual International Conference on Mobile Computing and Networking*, pages 108–116, 1996.
14. HP. Message Service. http://www.hpmiddleware.com/.
15. IBM. MQ Series. http://www.ibm.com/software/ts/mqseries/.
16. J. Jing, A. Helal, and A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, Vol.31(No.2), 1999.
17. C. Lau and A. Ryman. Developing XML Web services with WebSphere. *IBM System Journal*, Vol.41(No.2), 2002.
18. S. Lee, M. Gerla, and C. Chiang. On-Demand Multicast Routing Protocol. In *Proceedings of IEEE WCNC '99*, pages 1298–1302, September 1999.
19. J.C. Lin and S. Paul. Reliable Multicast Transport Protocol (RMTP). *Proceeding of IEEE INFOCOM '96*, pages 1414–1424., March 1996.

20. S. Maffeis. Middleware Support for Application-to-Application Wireless Messaging. White Paper by Softwired, 2000.
21. R. Monson-Haefel. *Java Message Service*. O'Reilly, 2001.
22. 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.
23. S. Ni, Y. Tseung, Y. Chen, and J. Sheu. The broasdcast problem in a mobile ad-hoc network. In *Proceedings of ACM/IEEE MobiCom*, August 1999.
24. S. Paul. *Multicasting on the Internet and Its Applications*. Kluwer, June 1998.
25. C. Perkins, E. Royer, and S. Das. *Ad-Hoc On-Demand Distance Vector(AODV) Routing*, June 1998. draft-ietf-manet-aodv-03.txt (work in progress).
26. RFC1112. *Host Extensions for IP Multicasting*. http://www.rfc-editor.org/rfc/rfc1112.txt.
27. RFC2581. *TCP Congestion Control*. http://www.rfc-editor.org/rfc/rfc2581.txt.
28. RFC3208. *PGM Reliable Transport Protocol Specification*. http://www.rfc-editor.org/rfc/rfc3208.txt.
29. RFC793. *Transmission Control Protocol*. http://www.rfc-editor.org/rfc/rfc793.txt.
30. L. Rizzo and L. Vicisano. RMDP: An FEC-based reliable multicast protocol for wireless environments. *ACM Mobile Computer and Communication Review*, 1998.
31. E. Royer and C. Perkins. *Multicast Ad-Hoc On-Demand Distance Vector (MAODV) Routing*, 2000. draft-ietf-manet-maodv-00.txt (work in progress).
32. Softwired. iBus Messaging. http://www.softwired-inc.com/.
33. Sun Microsystems. *Java Message Service (JMS) API Specification*. http://java.sun.com/products/jms/.
34. Sun Microsystems. *Java2 Platform Micro Edition (J2ME) specification*. http://sun.java.com/j2me/.
35. Sun Microsystems. *JCache: Java Temporary Caching API*. http://www.jcl.org/jsr/detail/107.prt.
36. Sun Microsystems. *RMI Profile Specification on Connected Device Configuration (CDC)*. http://java.sun.com/aboutjava/communityprocess/jsr/.
37. Talarian. Smart PGM. http://www.talarian.com/.
38. TIBCO. TIB/Rendezvous Concepts. http://www.rv.tibco.com.
39. G. Write and W. Stevens. *TCP/IP Illustrated*, volume 2. Addison-Wesley, 1994.