Abstract

A messaging proxy system is disclosed for the purpose of delivering data in the form of a portable message format from a producer running on a mobile or non-mobile computer, over any wireless network, by passing this data through an intermediary proxy computer program, to one or more recipients running on mobile or non-mobile computers. The system includes a message proxy computer program with at least one pluggable transport protocol adapter. The proxy contains a command subsystem for sending and receiving command- and message-tokens to and from the mobile clients. The system further includes a thin messaging middleware client to run on mobile devices. The thin messaging middleware client includes at least one pluggable protocol adapter. The client also comprises a command subsystem for sending and receiving command- and message-tokens to and from the proxy. The proxy also contains a communication subsystem for sending and receiving messages via a state of the art message oriented middleware.
A Mobile Device

**Thin Message Client Application**

1. TopicPublisher Pub = TopicSession.createPublisher(Topic T)

2. command token('Create a publisher', Topic T, Code P)

3. TopicPublisher Pub = TopicSession.createPublisher(Topic T)

4. Associate TopicPublisher “Pub” with Code “P”

5. Create a JSM message “msg”

6. Pub.publish(msg)

A Server Computer

**Thin Message Client Library**

7. command token('Do Publish', Code P, msg)

8. Get TopicPublisher “Pub” for code “P”

9. Pub.publish(msg)

Fig. 2
MESSAGING PROXY SYSTEM
CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a divisional of U.S. patent application Ser. No. 10/824,200, filed on Apr. 13, 2004, which is a continuation of U.S. patent application Ser. No. 09/611,629, filed on Jul. 7, 2000, now U.S. Pat. No. 6,721,779, each of which is incorporated herein by reference for all purposes.

FIELD OF THE INVENTION

[0002] The invention relates to techniques for the delivery of electronic messages between hardware or software components across wireless and wireline networks, between mobile and non-mobile devices.

BACKGROUND OF THE INVENTION

[0003] Message oriented middleware (MOM) has been available for many years. In October 1998, an industry standard emerged from Sun Microsystems, the Java Message Service (JMS). At a programming interface level, this standard describes how a messaging middleware is accessed from a Java application. The two main abstractions of JMS are “topics” (publish/subscribe messaging) and “queues” (point-to-point messaging). While the standard describes the interface to the messaging middleware, the implementation of the middleware is not specified. Also, integration of wireless mobile devices (such as phones, pagers, personal digital assistants or laptops) is not specified.

[0004] Existing messaging middleware allows one to access the middleware from non-mobile devices (personal computers or server computers) over wireline networks (Ethernet or Token Ring). These networks usually run communication protocols such as TCP/IP, HTTP or SSL. Supporting wireless mobile devices requires the vendor of the middleware to implement a message transmission protocol atop a wireless transport protocol (such as WAP, GSM, SMS, GPRS, or UMTS) and to integrate this message transmission protocol into the middleware.

[0005] This leads to limited applicability for the following reasons:

[0006] State of the art JMS messaging middleware requires more computer memory than is available on mobile devices.

[0007] Mobile devices, which are often disconnected from a corporate network, are unsupported in state of the art JMS messaging middleware products.

[0008] Wireless protocols such as WAP, SMS, GPRS or UMTS are not supported by state of the art JMS messaging middleware products, unless the TCP/IP, HTTP or SSL protocol is used atop those wireless protocols.

[0009] Though state of the art JMS messaging middleware products support communication protocols such as TCP/IP, HTTP, and SSL, they do not support any other communication protocols.

[0010] Further, there are considerable performance impacts, as TCP, HTTP or SSL were designed for wireline networks and thus do not perform well on wireless networks.

SUMMARY OF THE INVENTION

[0011] A first object of the invention is therefore to provide a system for the delivery of data between applications serving as clients and running on mobile wireless devices and applications running on computers of a wired network. Another object of the invention is to provide a method for delivering data between an application serving as client and running on a mobile wireless device and an application running on a computer of a wired network. Yet another object of the invention is to provide a computer program loadable into the memory of a computer usable for delivering messages between clients on mobile wireless devices and applications running on computers. A further object of the invention is to provide a computer program product comprising a computer usable medium having thereon computer readable program code means for implementing on a computer connected to a wired computer network. Still another object of the invention is to provide a computer program directly loadable into the memory of a mobile device and allowing the mobile device to access a messaging middleware product according to the state of the art, without needing to load that messaging middleware into the memory of the mobile device entirely.

[0012] The messaging proxy system outlined in this disclosure is a major technological advancement enabling users of state of the art messaging middleware products to send and receive messages to and from mobile devices, over any wireless transport protocol, without requiring that the state of the art messaging middleware be loaded into the memory of the mobile devices.

[0013] The system for running said message proxy installation includes a message proxy implemented by a computer program with a system architecture comprising at least one pluggable protocol adapter. In a preferred embodiment of the invention, the proxy further comprises at least one pluggable database adapter.

[0014] The invention also comprises a thin message client computer program directly loadable into the memory of a mobile device. This thin message client program allows a mobile device to exchange message and command tokens with a message oriented middleware according to the state of the art, by using the proxy computer program as an intermediary between the thin client and the message oriented middleware and hereby using at least one wireless transport protocol. The thin message client computer program embodies a system architecture of at least one pluggable protocol adapter. Preferably, it also embodies a system architecture of at least one pluggable database adapter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] In the following, an example of an embodiment of the invention is described with reference to drawings. In the drawings:

[0016] FIG. 1 provides a block diagram of a preferred embodiment of the system according to the present invention, and

[0017] FIG. 2 shows a UML sequence diagram of an embodiment of the method according to the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0018] Now with reference to the drawing, FIG. 1 provides a block diagram of a preferred embodiment of the present invention. It more particularly shows an installation of software tools loaded on non-mobile computers and mobile wireless devices, the installation comprising:
A message proxy 1.

Thin JMS message clients 2, 2', 2" linked to the proxy 1 with a wireless communication protocol.

JMS message oriented middleware 3 according to the state of the art, and

A JMS message oriented middleware client 4.

The block diagram is but one example of a message proxy infrastructure deployment. Any number of message proxies, thin JMS message clients, message oriented middleware products, and message oriented middleware product clients can be present in a specific installation.

The message proxy 1 may be implemented on a conventional computer network server, e.g. on a Windows-NT-server and may e.g. run in the background. It maintains client connections, maintains client subscriptions to JMS topics and queues, receives and forwards JMS messages, and stores JMS messages in its database, such that they will not be lost when a client is disconnected from the proxy.

The message proxy 1 comprises at least one pluggable transport protocol adapter. FIG. 1 shows an example of six specific wireless transport protocol adapters (WAP 1.0, UMTS 1.6, HTTP 1c, DAB/GSM Data 1d, SMS 1e, GPRS 1f). Any number of additional wireless protocol adapters 1g can be present. Pluggable protocol adapters allow the message proxy to send and receive messages to and from message clients using arbitrary wireless protocols. A protocol adapter embodies an existing transport protocol, such as GPRS or TCP/IP, and also provides additional features on top of the existing transport protocol. Examples of such additional features include data encryption and guaranteed delivery of messages. A protocol adapter is divided into one or more protocol objects. Each protocol object provides one part of the functionality offered by the protocol adapter. For example, a protocol object can encrypt data, or compress data, or request the sender of the data to retransmit a message which was lost on the network.

The message proxy 1 also comprises a database adapter. This allows the proxy to store messages and client subscription information into arbitrary databases.

On startup, the message proxy 1 reads its configuration data and initializes all configured protocol adapters. It also initializes the topic and queue subscriptions of all the message clients which are known to the proxy. At runtime, additional protocol adapters can be started, or running protocol adapters can be stopped, without interrupting the messaging proxy service (however, if a specific protocol adapter is stopped, service over this adapter is no longer available). At runtime, additional clients can be connected to the proxy, or existing clients can be disconnected from the proxy.

Each thin JMS message client 2, 2', 2" is installed on a mobile wireless device such as a mobilephone, a small laptop computer with a wireless modem, a palmtop device or any other device comprising a processor, a memory and communication means for communicating wirelessly. It contains a JMS programming library which is identical or similar to at least a part of the programming library used by messaging middle 3 of the state of the art. The thin JMS message client library is small enough to be loaded into the memory of the mobile devices which have constrained memory and processing power.

Such a small footprint of the thin JMS message client library is achieved by offloading to the client to the proxy most of the computations and most of the state information which a JMS client application ought to perform or to maintain. The thin JMS message client mainly consists of the JMS interface. Most of the Java code necessary for implementing the interface is running on the proxy, and not on the thin JMS message client. The proxy also maintains the JMS state information associated with the client. For example, the proxy stores the JMS messages which have not been acknowledged by the client yet. Also, the thin JMS message client does not need to store the names of the queues and topics it is subscribed to. This information is stored only by the proxy. Internally, the thin JMS client uses code information, such as number values, to refer to topics and queues. This code information can be as small as one byte. The actual representation of those queues and topics, which can be hundreds or thousands of bytes for each topic or queue, is contained in the proxy. When the thin JMS client wishes to publish a message on a certain topic, the client sends to the proxy only the JMS message and the code information related with the topic. All this considerably reduces the footprint of the thin JMS client.

The thin JMS message client 2 also contains a command and message transmission system comprising a transport protocol adapter 2a, 2'a, 2"a used for informing the proxy of what JMS topics and queues the client wants to subscribe to.

The message client 2, 2', 2" also comprises a database adapter. This allows the client to store JMS messages and other information locally, using arbitrary databases. The message database is necessary to ensure that JMS messages and JMS subscriptions submitted by the client are not lost in case the client cannot communicate with the proxy due to lack of wireless network coverage, or because the proxy is not running.

A message client 2, 2', 2" links to the message proxy 1, using its transport protocol adapter 2a, 2'a, 2"a. If a matching protocol adapter is running on the proxy, the connection is successful. Further communication between message client and message proxy is according to the familiar publish/subscribe or point-to-point model of JMS.

JMS topics or JMS queues are named and administered independently of the protocol adapters involved. If a client connects to the proxy using the “WAP” protocol adapter, it can communicate with a client that connected using the “GPRS” protocol adapter, if both clients use the same JMS topic or queue.

The protocol adapters encapsulate at least one logic needed to:

Interface with a transport protocol, such as HTTP, WAP or GSM Data.

Specify and guarantee a quality of service for the message delivery.

Certain transport protocols operate in a “best effort” delivery mode. Thus, simply adapting to a specific protocol is not always enough (unless “best effort” is the desired message delivery guarantee). Thus, protocol adapters consist of both the transport protocol mechanism, and a quality of service mechanism to improve the basic network delivery guarantee.

Network reliability is improved as follows. The sending protocol adapter attaches a reliability indicator such as a sequence number to all outgoing messages. The reliability indicator is varied in a predefined manner upon sending a message. E.g., the sequence number is incremented by one after each sent message. The receiver application uses the reliability indicator of the incoming message to detect
whether a message was lost. In the described example, this is the
case when the sequence number of the just received mes-
gage is greater than the sequence number of the previous
message, plus one. In the event of message loss, the receiver
sends a command token to the sender indicating which mes-
gages are to be retransmitted. The sender then retransmits
the requested messages. The sender keeps messages in a local
database to be able to fulfill a message retransmission request.

[0039] The database adapters encapsulate at least one logic
needed to:

[0040] Interface with a database product, such as Point-
Base, Oracle, DB/2, or Sybase, OR to interface with a
portable database access software such as JDBC or ODBC.

[0041] Store and retrieve JMS messages and JMS subscrip-
tion requests.

[0042] The message client 2 implements e.g. the JMS API
from Sun Microsystems. It cooperates with the proxy to
achieve full JMS functionality. When the client wants to
subscribe to a JMS queue or topic, its command subsystem
creates a command token containing the subscription in-
formation. The command token is then sent to the proxy using
wireless communication. To this end, the token is sent via a
protocol adapter 2a, 2b, 2c at the client side and received by
a protocol adapter 1a, 1b, 1c, 1d, 1e, 1f, or 1g at the proxy side.

[0043] On receipt of a command token, the proxy 1 reads
the subscription information contained in the token, and per-
forms a JMS subscription with the state of the art middle-
ware, on behalf of the client.

[0044] Further command tokens are generated when the
client wants to unsubscribe from a JMS topic or queue, when
the client wants to transmit a JMS message, or for any other
JMS action which is requested by the client.

[0045] When a JMS message is received on a topic or queue
the proxy 1 is subscribed to on behalf of the client, the proxy
creates a message token containing the data of the JMS
message. The message token is then sent to the client 2 using
wireless communication. For that the token is sent via a
protocol adapter 1a, 1b, 1c, 1d, 1e, 1f, or 1g at the proxy side,
and received by the protocol adapter 2a, 2b, 2c at the client side.

[0046] On receipt of such a message token by a thin JMS
message client 2, a JMS message is created by the client.
Then, the JMS message is processed by the client. For
example, the message can be visualized in a graphical user
interface.

[0047] The JMS message oriented middleware 3 according
to the state of the art can be any JMS messaging middleware
product, for example, IBM’s MQSeries, SoftWired’s iBus, or
Progress’ SonicMQ.

[0048] The JMS message oriented middleware client 4 is a
client application implemented on a non-mobile computer,
also on a computer network connected to a wired computer network,
using a state of the art JMS messaging middleware 3. One or
more JMS message oriented middleware clients 4 according
to the state of the art can be present.

[0049] For describing an example of the communication
between different examples, it is assumed that the thin
JMS message client 2 is subscribed to a topic T. Such a topic
T can, depending on the application, denote a stream of stock
quotes, of sports news, or denote a transmission channel
carrying digital audio. When a state of the art JMS message
oriented middleware client 4 sends a JMS message to topic T,
the message is passed first to the state of the art JMS message
oriented middleware 3. The message will then be received by
the proxy 1 on behalf of thin client 2. Next, proxy 1 transmits
the JMS message to client 2 in the form of a message token
using one of its transport protocol adapters 1a, 1b, 1c, 1d, 1e,
1f, or 1g. Finally, client 2 receives the JMS message on topic
T as if it was accessing the state of the art JMS message
oriented middleware 3 directly.

[0050] In order to show this procedure in more detail, in
the following an example of the method for delivering information
between applications running on mobile wireless devices
and serving as clients and applications running on non-mobile
computers is described with reference to FIG. 2.

[0051] The sequence diagram of FIG. 2 shows the interac-
tions—represented by arrows—occurring between a mobile
client and a proxy during one information exchange, namely
during the creation of a JMS TopicPublisher for a topic T by
the mobile client and a subsequent publishing of a message
published on topic T. In the diagram, the mobile client is
symbolized by a shaded and dashed box. The vertical line on
the right hand side of the figure represents the message proxy.
The method steps are denoted by numbers which are not to be
confused with the reference numerals of FIG. 1.

[0052] 1. A JMS TopicPublisher object “Pub” is created,
on request of the application, for JMS publish subscribe
topic “T”. Later, “Pub” will allow the mobile client application
to publish a JMS message on topic “T”.

[0053] 2. The Thin Message Client Library creates a com-
mand token containing the information which is needed by
the proxy for allocating a JMS TopicPublisher, on behalf of
the client. The command token contains a code information
(e.g. a one-byte number) denoting a ‘Create a publisher’
command. It also contains the JMS topic “T” the publisher
shall be tied to (e.g. a one-byte number), as well as an in-
formation Code “P” (e.g. a one-byte number) denoting the
publisher.

[0054] 3. The proxy creates a JMS TopicPublisher “Pub” for
topic “T” on behalf of the thin client.

[0055] 4. The proxy associates TopicPublisher “Pub” with
the code information “P”. This can be done by storing the
TopicPublisher “Pub” into a dictionary, using code infor-
mation “P” as the search key.

[0056] 5. The client application creates a JMS message
“msg” containing application specific information, e.g.,
a book order. This step, of course, as well as the subsequent
steps 6, can be carried out following to or simultaneously to steps 3
and 4.

[0057] 6. The client application now publishes JMS mes-
gage “msg” on topic “T”, using TopicPublisher “Pub”.

[0058] 7. The Thin Message Client Library creates a com-
mand token containing the information which is needed by
the proxy for publishing the message using state-of-the-art
JMS middleware. The command token contains a code infor-
mation (e.g. a one-byte number) denoting a ‘Do publish’
command. It also contains the code information for the Top-
icPublisher (Code “P”) as well as the message “msg”.

[0059] 8. The proxy retrieves the TopicPublisher “Pub”,
which is associated with Code “P”. This publisher “pub” was

[0060] 9. Finally, the proxy publishes the JMS message
“msg” on topic “T” using a state of the art JMS middle-
ware. Concretely, the proxy forwards the message “msg” on topic
“T” to a state of the art JMS application using JMS.

GLOSSARY OF TERMS USED

[0061] TCP: Transmission Control Protocol

[0062] IP: Internet Protocol
HTTP: Hypertext Transfer Protocol
WAP: Wireless Application Protocol
WDTP: WAP Wireless Datagram Protocol
SSL: Secure Socket Layer
JMS: Java Message Service (http://java.sun.com/products/jms/)
PDA: Personal Digital Assistant
SMS: Short Messaging Service
GSM: Global System for Mobile Telecommunication
DAB: Digital Audio Broadcast
JDBC: Java Database Connectivity (http://java.sun.com/products/jdbc/)
ODBC: Microsoft's Open Database Connectivity
MOM: Message Oriented Middleware

What is claimed is:
1. A method for delivering data between an application serving as client and running on a mobile wireless device and a message oriented middleware application running on a computer of a wired network,
   comprising the steps of
   implementing, on the mobile wireless device, a messaging system Application Programming Interface (API);
   running a message proxy on a computer connected to the wired network, the proxy maintaining client subscriptions to topics and queues, and receiving and forwarding messages;
   and at least one of the following two steps of
   the client calling a command provided by the messaging system API;
   the proxy receiving a message on behalf of the client;
   and further comprising the step of
   the proxy acting towards the message oriented middleware on behalf of the client as if the client was accessing the message oriented middleware directly.
2. The method of claim 1, wherein a thin message client is installed on the mobile wireless device
   and implements the API of the messaging system,
   comprising the steps of
   the client calling a command provided by the thin message client API;
   a command subsystem of the thin message client creating a code information representing the command;
   sending the code information to the message proxy using wireless communication;
   simultaneously or subsequently, transmitting message data to the message proxy;
   the message proxy maintaining message system state information associated with the client;
   creating, by the message proxy, at least one MOM command referred to by the code information; and
   the message proxy forwarding the message data to the message oriented middleware application, using said at least one MOM command.
3. The method of claim 1, wherein a thin message client is installed on the mobile wireless device
   and implements the API of the messaging system,
   comprising the steps of
   the proxy receiving a message on behalf of the client;
   the proxy creating a code information representing the message;
   the proxy sending the code information to the thin message client using wireless communication;
   the thin message client creating, from the code information, a message; and
   the client processing the message.
4. The method of claim 1, comprising the step of
   at least one of the clients and the proxy storing messages in a local database, such that they will not be lost when a client is disconnected from the proxy.
5. The method of claim 1, comprising the steps of
   the message proxy starting, at runtime, a pluggable transport protocol adapter enabling communication over a given transport protocol;
   the message proxy sending and receiving messages to and from message clients by means of the transport protocol adapter, using wireless protocols.
6. The method of claim 5, wherein said at least one transport protocol adapter supports HTTP or SMS or WAP or WDP or GPRS or UMTS.
7. The method of claim 2, wherein the command provided by the thin message client API and represented by code information or tokens is one of "subscribe", "unsubscribe", "create topic", "publish".
8. The method of claim 1, wherein the API client library is a JMS (Java Messaging System) library.
9. A system for delivering data between an application serving as client and running on a mobile wireless device and a message oriented middleware application running on a computer of a wired network,
   comprising
   a messaging system Application Programming Interface (API) implemented on the mobile wireless device;
   a message proxy implemented on a computer connected to the wired network, the proxy being configured to maintain client subscriptions to topics and queues, and to receive and forward messages;
   the proxy being configured to act towards the message oriented middleware on behalf of the client as if the client was accessing the message oriented middleware directly;
   in at least one of the cases of
   the client calling a command provided by the messaging system API;
   the proxy receiving a message on behalf of the client.
10. The system of claim 9, wherein the message proxy comprises a wireless transport protocol adapter implemented before start-up of the message proxy and a wireless transport protocol adapter implemented by a program code at run-time of the message proxy.

* * * * *