INTERWORKS 2000

MESSAGING ACROSS PLATFORMS

ASYNCHRONOUS APPLICATION INTEGRATION ACROSS DISPARATE PLATFORMS

Nayan B. Ruparelia • RupSoft Ltd. • 43 Haverford Way • Edgware • Middx. • HA8 6DJ • England
Tel.: +44-(0)20-8952 1293 • Fax: +44-(0)20-8952 1672 • Email: nayan@acm.org
www.navadvipa.fsnet.co.uk

MESSAGING ACROSS PLATFORMS

ASYNCHRONOUS APPLICATION INTEGRATION ACROSS DISPARATE PLATFORMS

ABSTRACT

Middleware allows applications running on various operating systems like Linux, Windows 2000 and HP-UX to communicate with each other. Such communication is de-coupled. This means that the receiving application need not be available when the sender transmits a message. The principle is rather similar to an email being sent by one person to another -- in the sense that the recipient of the email need not be available when the email is sent. One major difference, however, is that the address of the message is not specified when using middleware. The receiving application obtains its message from its message queue. There are three different paradigms that currently exist: i) asynchronous messaging, ii) publish-subscribe messaging and iii) content-driven messaging. In the first case, message queues are used so that messages between applications are queued. A queue manager effects the transfer of messages to the correct recipient. In the second case, the subscriber makes known its interest in a particular topic, and it consequently receives messages that are published on those topics of interest to it. A message broker usually actuates the transfer of messages as it keeps track of the various publishers publishing under their topics and subscribers subscribing to topics of interest. In the final case, a rules engine evaluates various rules under which messages are routed to their destination. In such cases, messages in any format may be sent from one application to another, and the reformatting can be done on the fly by the rules engine as it routes the messages to their intended destination. The presentation shall consider these three types of messaging, and discuss the advantages and disadvantages of each. A case study, using IBM's MQSeries Integrator shall be considered where the issues pertaining to the architecture of the system, nomenclature, security issues and message formats shall be dealt with from a practical perspective. This should give a 'hands on' appreciation to the issues at hand when commissioning a middleware system.

INTRODUCTION

Middleware is a software layer that exists between an operating system and its applications. It provides transparency between operating systems so that applications on different operating systems may communicate with each other. This communication is effected using three paradigms: a) asynchronous messaging, b) publish-subscribe messaging, and c) content-driven messaging. In the first case, message queues are used so that messages between applications are queued. A queue manager effects the transfer of messages to the correct recipient. In the second case, the subscriber makes known its interest in a particular topic, and it consequently receives messages that are published on those topics of interest to it. A message broker usually actuates the transfer of messages as it keeps track of the various publishers publishing under their topics and subscribers subscribing to topics of interest. In the final case, a rules engine evaluates various rules under which messages are routed to their destination. The messages routed by the rules engine can be in any format. The rules engine invokes the formatter, which reformats the messages from the source format to the required destination format. The formatting is performed on the fly by the rules engine. This form of messaging is also known as content-driven messaging.

In middleware terms, the destination queue or message subscriber is known as a 'consumer' and the source queue or message publisher is known as the 'producer'. Some ground-rules exist for the effective transmission of messages between producers and consumers. These are as follows.

a) De-coupling between producers and consumers

Message delivery is not affected if the consumer is unavailable at the time the producers sends or publishes a message.

b) Guaranteed delivery of messages

The message needs to be delivered to the correct recipient on a guaranteed basis. Not all middleware technologies guarantee the physical delivery of a message should the middleware's message broker be unavailable.

c) Prioritisation

A mechanism should exist for the prioritisation of messages. There is, of course, a distinction between the priority attached to a message and the security with which it is required to be delivered.

d) Security

There should be a capability for messages to be secure whilst being transmitted, and for them to be made available only to the intended recipient.

Situations abound where middleware plays an integral part of an enterprise's infrastructure. Its primary uses are in organisations that have a number of disparate operating systems or platforms, and the need exists to integrate these such that information can be passed freely between the various systems. The systems can vary from legacy systems, mainframes, workstations, LANs, WANS and personal computers. For example, an investment bank might store its financial records on a OS/390 mainframe and it may have a number of LINUX servers that service the day-to-day business of the institution by hosting an Oracle database in which the daily trading information is stored. In addition, there could be Windows 2000 workstations that run a number of applications like spreadsheets that the traders use for trading securities. In order

for the most recent data to be made available to all parties, regardless of the system they are using in the organisation, middleware could be used, as Figure 1 shows. Queues take data from the various applications or databases in Figure 1 and the rules engine of the middleware layer makes the data available at destination queues after formatting the data to suit the format required by the destination application. Thus, EDI format data stored on the mainframe could be transformed to XML on the HP-UX or Linux server and vice versa. Similarly, comma-delimited ASCII data stored by the spreadsheet could be transformed to either XML (for the Linux or HP-UX server) or EDI. This formatting is performed on the fly as and when the rules engine routes the messages to the relevant destination queues.

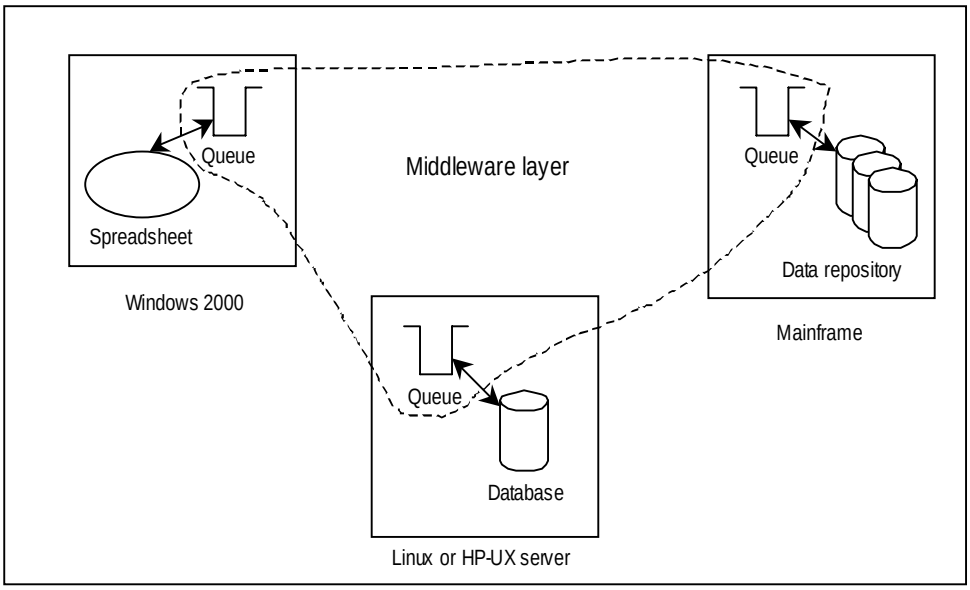

Figure 1: Example of the use of middleware

The following middleware technologies are currently available:

- MQSeries family

  IBM provides the core MQSeries product, which provides messaging and queuing. MQSeries is extended by the incorporation of a message formatter (which has extensive formatting capabilities from any user-defined complex format to another) and a rules engine. This extended product is known as MQSeries Integrator. As an addition to MQSeries, IBM also provides MQSeries Publish-Subscribe as an add-in. This allows messages to be broadcast under relevant topics that consumers can subscribe to.

- TIB/Rendezvous

  TIBCO is the original Middleware Company. Its main offering is TIB/Rendezvous, which uses the publish-subscribe messaging paradigm.

- NEONet

  NEONet is a product of NEON Inc. It consists of a messaging and queuing layer, a message formatter and a rules engine which routes messages from to destination queues on the basis of their content, as defined by user-defined rules. The NEON formatter and rules engine is licensed to IBM for use in MQSeries Integrator.

- CORBA ORB

  The Common Object Request Broker Architecture is much more than middleware. It is actually an architecture that allows distributed object management. The broker, or ORB (Object Request Broker), of CORBA is a proxy that de-couples objects from their clients. The ORB can therefore be configured to play the role of a message broker. It, however, is a layer lower -- and closer to the operating system -- than the other middleware technologies that we have considered. In this regard, CORBA is not middleware. However, its object broker can be configured to be so by suitable programming constructs.

There is much research being performed on middleware. For instance, none of the major middleware systems currently offers a rules-based publish-subscribe technology. This is when a publisher only publishes a certain topic if various criteria defined by the subscribers to that topic are met. Enhancements such as these are currently being developed. The following are some of the research projects currently being undertaken on middleware.

GRYPHON

Gryphon is a robust publish/subscribe message broker being developed by IBM's Watson Research Centre (gryphon@watson.ibm.com). It provides content-based publish/subscribe functionality and offers scalability over geographical regions. It also uses some advanced techniques to automatically reconfigure itself should a broker fail so that the message traffic can be re-routed via other brokers. Gryphon also supports four authentication mechanisms for verifying client identity: simple password authorisation, mutual secure password authentication (password is never sent over the wire), asymmetric SSL (password sent over a secure SSL connection to the server) and symmetric SSL (both client and server use certificates to authenticate each other). For development at the client side, the Java Messaging Service (JMS) API is supported. Native C/C++ interfacing for mission and performance critical applications is

also supported. At present, Gryphon supports non-blocking I/O socket libraries for Windows 2000, Linux, and AIX.

IPROXY

*i*PROXY has been developed by AT&T Research (iproxy@research.att.com) as middleware specifically for applications that access Web services, such as browsers, search engines, virtual machines, indexing tools, intelligent agents, and Intranet applications. It allows accessing, caching, and processing of web data. Although many internet browsers like Netscape, Microsoft IE, and Mosaic provide suitable facilities for searching and retrieving information on the network in a convenient and productive manner for end-users, they do not offer open architectures so that other applications can access functions available inside the browser. As a result, no general way currently exists for applications (including browsers) to share caches. Additionally, it is very difficult to customise most browser features such as having the choice for different proxies for different domains, turning caches on and off, and clearing cached data for selected web-sites. *i*PROXY addresses these issues by introducing a new middleware for browsers and other Web application programs. It provides an API (application programming interface) for retrieving Web data, processing it, and accepting notifications from remote servers. *i*PROXY has a built-in web server, which transforms the client/server communication model used by the WWW consortium to a peer-to-peer communication model. This enables end-users to download home-page sets, customise browser features, maintain caches and customise the URL name space on their local machines. This is in addition to serving as a middleware layer for applications across disparate platforms in an Intranet. *i*PROXY is designed to run under Windows and UNIX-like operating systems and it remains transparent to existing browsers -- which treat it like a regular proxy server.

RESOURCES

Below is a list of resources that can be helpful for obtaining further information on specific middleware technologies.

WEB-SITES

http://www.eaijournal.com

http://www.middleware.org

http://www.navadvipa.fsnet.co.uk

http://www.neonsoft.com

http://www-4.ibm.com/software/ts/mqseries

BOOKS

The following are some of the books available on middleware. CORBA has a large selection of books available for it, whereas very few books exist for the other middleware technologies.

CORBA

Corba Distributed Objects : Using Orbix
by Sean Baker
536 pages (November 1997)
Addison-Wesley Pub Co; ISBN: 0201924757

The Corba Reference Guide
by Alan Pope
407 pages (January 1998)
Addison-Wesley Pub Co; ISBN: 0201633868

Enterprise Application Integration with CORBA Component and Web-Based Solutions
by Ron Zahavi
560 pages (November 1999)
John Wiley & Sons; ISBN: 0471327204

MQSERIES

Distributed Computing with IBM(r) MQSeries
by Leonard Gilman, Richard Schreiber and Len Gilman
283 pages (October 1996)
John Wiley & Sons; ISBN: 0471149349

MQSeries Messaging
by Nayan Ruparelia
550 pages (August 2000)
Manning Publications.

ABOUT THE PRESENTER

Nayan B. Ruparelia obtained his B.Sc. degree in 1984 from London University (King's College) in Electronics Engineering. Subsequently, in 1987, he obtained his M.Sc. in Digital Systems & Instrumentation from Westminster University with sponsorship from Thorn EMI Datatech, where he worked as an Electronics Engineer. From 1988 to 1994, he worked at Dowty Controls as a Principal Electronics Engineer where he designed and researched various avionics systems for aircraft guidance and control. He represented Dowty Controls at various forums including the ASSC Data Transmission and Fibre Optics work-groups. These two work-groups were directed by the Ministry of Defence for the purpose of establishing data transmission standards and requirements that would later be implemented as standard specifications for use by the entire Aerospace industry across Europe. From 1994 onwards, he has been working as an independent I.T. consultant and, from 1997 to 1999, he worked as a sub-contractor for NEON, a NASDAQ-listed company which provides middle-ware products and consultancy services. He joined NEON at a time when he was the only technical consultant in its UK office -- which had a head-count of less than ten. In 1999, the UK subsidiary alone had garnered sales revenues in excess of £16 million and had a head-count approaching five hundred. As a NEON consultant, he obtained experience on various operating systems with the NEONet formatter and rules engine (which were both developed by NEON and are used as essential components in IBM's MQSeries Integrator). Nayan has written a book, entitled MQSeries Messaging, which shall be published in summer 2000 by Manning Publications (www.manning.com). Currently, he continues to work as an independent consultant on TIBCO, NEONet and MQSeries. Nayan is a member of the Association of Computer Machinery (www.acm.org).