A JMS Message Transport Protocol for the JADE Platform

Edward Curry, Desmond Chambers and Gerard Lyons
Department of Information Technology,
National University of Ireland, Galway, Ireland.
{edward.curry, des.chambers, gerard.lyons}@nuigalway.ie

Abstract

A prerequisite of joining an enterprise system is the ability to cope with the rigorous demands experienced within such systems. One of the most fundamental of these demands is the requirement for enterprise-level systems to have guaranteed reliable messaging between the participants of the system. Our research involves integrating an agent platform with an enterprise messaging service. This first step in combining agent technology with a mainstream messaging service is vital to the participation of agent systems within the digital enterprise. This paper introduces a new Message Transport Protocol (MTP) for the Java Agent DEvelopment (JADE) platform. The new protocol uses the Java Messaging Service (JMS) to deliver inter-platform communication between agent platforms. The paper provides a brief overview of the design of this new MTP, evaluates its performance, and examines the benefits of the MTP in comparison to the other available MTPs, it then concludes and highlights plans for the development of the MTP.

1. Introduction

The Foundation for Intelligent Physical Agents (FIPA) is an international organisation that is dedicated to promoting the industry of intelligent agents by openly developing specifications to support interoperability amongst agents and agent-based systems. FIPA require that a Message Transport Service (MTS) [1] must provide inter-platform agent-based communication. This MTS is used to send and receive FIPA compliant ACL messages to and from remote agents. The MTS can use a number of different Message Transport Protocols (MTP) to handle the physical delivery of the messages, current MTPs include HTTP [2], WAP [3] and IIOP [4].

A basic requirement for enterprise-level systems is a need for guaranteed reliable messaging between participants of the system, these messaging requirements are provided by Enterprise-Messaging Services (EMS) or Message-Orientated Middleware (MOM). Domain experts in messaging such as IBM, Sun and Sonic Software implement such messaging services. Deployable as a standalone server or integrated into an application server, EMS/MOM provides reliable messaging and decoupling for large-scale systems, more advanced high-end commercial implementations ease integration issues and provide massive-scalability with clustering support to avoid a single point of failure.

This work is being conducted within the scope of the Virtual Logistics multi-Agent Broker (V-LAB) research project at the Department of Information Technology, National University of Ireland, Galway. V-LAB uses the Java Agent DEvelopment (JADE) platform as its agent's platform. JADE [5] is a FIPA compliant software framework designed to assist the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. JADE is designed to simplify the development of a software agent while ensuring standard compliance through a comprehensive set of system services and agents.

Researchers have noted that no agent platform implementation to date has reached the quality of commercial software applications [6]. A central theme of our research is to integrate the JADE agent platform with an EMS/MOM in order to allow JADE agents to participate within an enterprise system. Development has been completed on 1.0 version of the Java Messaging Service-Message Transport Protocol (JMS-MTP) and this has been released under the LGPL open source license, available for download from the JADE website or from http://ecrg.it.nuigalway.ie/jade.

The remainder of this paper gives an overview of the JADE messaging architecture and EMS/MOM. The paper also covers the design, implementation and evaluation of the JADE JMS-MTP with conclusions and future development plans.

2. JADE messaging

JADE is a FIPA compliant agent platform and development framework. Agent platforms are responsible for dealing with agent services such as messaging (transport, encoding and parsing), scheduling, agent lifecycle management and other common resources. JADE agents are executed in a container where they share these services with other agents present in the container. Our main interest in the JADE platform concerns its messaging capabilities.

2.1. JADE messaging architectural overview

Messaging within the JADE agent platform falls into two categories, *intra*-platform and *inter*-platform agent messaging. *Intra*-platform messaging takes places when two or more agents that resided in the same platform wish to communicate, these communications take place over JADEs Internal Message Transport Protocol (IMTP). IMTP techniques currently used by JADE are Java Event passing (passing memory locations, very efficient) for intra-container messaging and Remote Method Invocation (RMI) for inter-container messaging. Our research focuses on inter-platform communications.

2.2. Inter-platform communications

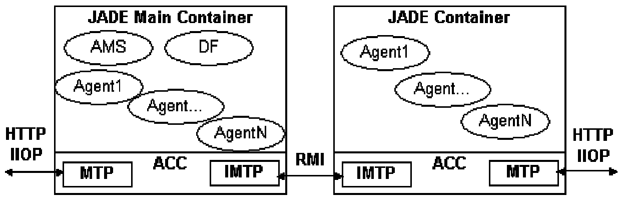

Figure 1. JADE Messaging Architecture

Figure 1 further illustrates the role of the IMTP and MTP within JADEs messaging architecture. *Inter*-platform communication takes place through the Agent Communication Channel (ACC), which can use a number of MTPs to transport agent messages to external platforms. As of version 2.1, JADE has been designed to allow ‘plugable’ MTP implementations; achieved by exposing a number of Java interfaces in the jade.mtp package [7].

There are a number of MTPs currently available for the JADE platform CORBA IIOP based on the Sun ORB (default MTP that ships with JADE), CORBA IIOP based on the ORBACUS ORB [8] and a HTTP MTP [9].

2.3. Messaging-orientated middleware

As enterprises continue to deploy distributed applications on ever increasing scales, transcending geographical, organisational and traditional commercial boundaries, the demands placed upon their communication and transaction infrastructures increases exponentially. Upon examination, direct RMI or Remote Procedure Call (RPC) mechanisms quickly fail to meet the challenges presented in such environments.

In order to cope with the demands of these systems, an alternative to the RPC mechanism of distribution has emerged. This new mechanism called Message-Orientated Middleware or MOM provides a clean method of communication between disparate software entities. MOM systems are one of the cornerstone foundations that distributed enterprise systems are built upon. MOM can be defined as any middleware infrastructure that provides messaging capabilities; in the context of this paper, we use the term MOM to define Enterprise-Messaging Services (EMS) such as, WebSphere MQ, SonicMQ, etc. These services provide the backbone infrastructure to create cohesive distributed applications.

The main benefits of MOM come from its asynchronous interaction model and its use of message queues. These queues allow each participating system to proceed at its own pace without interruption. MOM introduces transactional capability and a high Quality of Service (QoS) not found in the RPC model, it also provides a number of messaging models to solve a variety of different messaging challenges. The process of building dynamic, highly flexible, cohesive enterprise-class distributed systems can be simplified by utilising an EMS/MOM as the communications backbone.

2.4. Java messaging service

The Java Message Service (JMS) [10] specification provides an API and a set of semantics that describe the interface and general behaviour of an EMS. The goal of the JMS specification is to provide a universal way to interact with multiple heterogeneous EMS in a consistent manner. An application using the JMS API is able to plug-in any JMS compatible EMS, this allows the developer to choose the best EMS implementation for the applications requirements.

3. The JMS-MTP

The goal of this research is to implement a Java Messaging Service (JMS) Message Transport Protocol (MTP) for the JADE agent platform; this new transportation plug-in is called the JMS-MTP. The remainder of this section examines its design and highlights some features of the new MTP.

3.1. JMS-MTP overview

The premise of the JMS-MTP is based on the notion that inter-platform messages for a JADE platform are enhanced by sending them to external entities using an EMS as the delivery and decoupling mechanism. Areas of particular interest in the MTPs design include, the queue structure on the JMS provider, the encoding formats of the messages and support for multiple JMS providers.

3.2. Queue structure

The JMS-MTP utilises a queue structure, as illustrated in Figure 2, which requires each platform to have its own queue for incoming messages. Only one platform may read messages from a queue, however a platform may have multiple queues on a single provider as a load balancing mechanism, or could have multiple queues on different providers.

To avoid naming conflicts we prefer for all JADE platform queues to be placed under the '/jade' prefix and for each JADE platform to listen to a queue with the same name as its host. For example, a platform with a hostname of 'frodo' would listen to a queue called '/jade/frodo' while a platform with a host of 'IBM' would listen to a queue of name '/jade/IBM'.

When a platform connects to a JMS server they are required to check for the existence of a queue for their host, if one exists they should connect to it and start listening, if a queue does not exist they are required to connect to the JMS provider's administration interface and create a new queue for the platform.

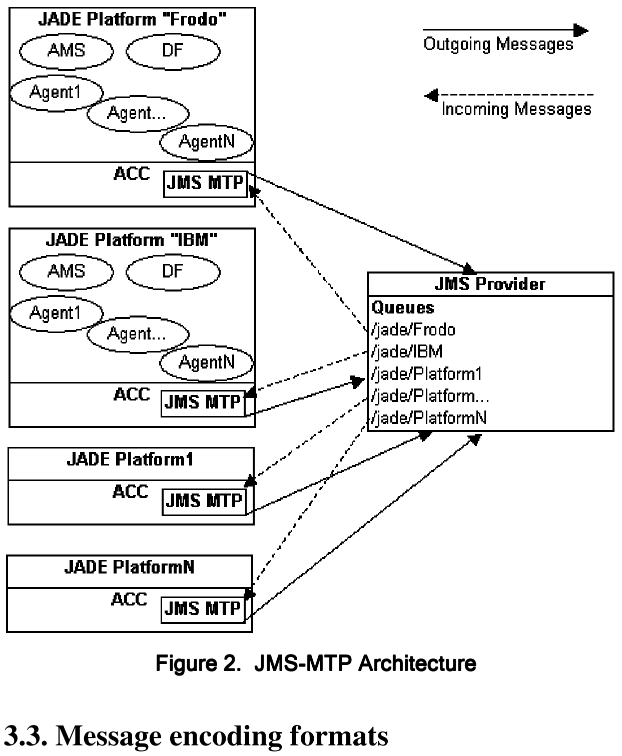

Figure 2. JMS-MTP Architecture

3.3. Message encoding formats

Messages sent with the JMS-MTP can be in two formats, using FIPA compatible XML in a JMS TextMessage or with a custom JMS-MTP MapMessage.

Messages encoded as XML are converted to the FIPA Agent Message Transport Envelope Representation in XML Specification [11]. Messages are encoded and decoded using the JDOM framework, with Apaches Xerces as the default SAX parser and delivered using a JMS TextMessage. The overhead of parsing XML may elongate the time required for the agent platform to process the message, thereby limiting the overall message throughput of the MTP.

In many respects, the MapMessage has all the same virtues as XML without the performance hit of text parsing. With regard to portability, most JMS vendors will automatically convert a MapMessage produced by a Java application to a non-Java environment. Hence, the MapMessage is an alternative to using the XML format.

However, currently there is no FIPA specification for such a hash format, thus usage of this format is limited to scenarios where both sides of the communication are performed using the JMS-MTP. It is our hope that FIPA will address this issue by defining a standard specification for a HashTable-like message format; our implementation, using the JMS MapMessage, may provide a good foundation for this standard.

3.4. Supported JMS providers

The fundamental concept of JMS is to provide a universal interface to proprietary vendor specific EMSs. Numerous vendors currently ship JMS compatible MOM products; these include IBM's WebSphere MQ (formerly MQSeries), Sun's JMQ, Sonic Software's SonicMQ and JBoss Group's JBossMQ.

The JMS specification provides a consistent API set that gives access to the common features of an EMS. However, it does not define every aspect of a service, such as the administration interface of a messaging service. Hence service management and configuration is still vendor specific such as, the dynamic creation of a queue at runtime. Since the JMS-MTP relies on the ability of a platform to create a new queue on a provider at runtime, we have exposed an interface to allow new JMS providers, with their vendor specific code, to be easily 'plugable' into the JMS-MTP. Once this interface has been implemented, the JMS-MTP can dynamically load the new provider at runtime, allowing for simultaneous usage of multiple JMS compatible EMS providers. The JMS-MTP currently ships with support for the JBossMQ [12] and SonicMQ [13] JMS compatible EMSs.

3.5. Configurability of JMS-MTP

A number of settings that controls the JMS-MTP are configurable, these include the default setting for the XML parser, default platform address and any JMS provider specific configurations. The Quality of Service (QoS) level may also be set with the option for persistent and non-persistent messages. When messages are marked as persistent, it is the responsibility of the JMS provider to guarantee that the message will never be lost. In order to fulfil this responsibility the JMS provider must store the message in a non-volatile medium such as a hard disk or solid-state ram.

4. Research evaluation

In order to evaluate this new MTP, it shall be compared against other available MTPs for performance and scalability in a message throughput environment. The JMS-MTP also provides a number of unique benefits for the JADE agent platform. These benefits will be highlighted.

4.1. Performance and scalability

In order to compare the JMS-MTP to other JADE MTPs in performance and scalability we performed benchmarking tests. These tests were carried out using the test-suite [14] for JADE MTPs developed by Telecom Italia Lab (TILAB). This test-suite consists of a number of sender and receiver agent couples; these agents measure the average roundtrip time (avgRTT) for a circular exchange of an ACL message between a sender agent and a receiver agent. The test-bed assembled to run the benchmarks is of comparable performance to that used by TILAB in their own benchmarks tests [14]. Running their tests on our test-bed produced similar results.

4.2. Benchmarking results

The benchmark results, presented in Figure 3, show that when used with its custom MapMessage format the JMS-MTP outperforms all other currently available JADE MTPs in tests run from 1 to 100 agent couples. When used in conjunction with FIPA compliant XML, the JMS-MTP is comparable to the performance of the HTTP MTP even though the JMS-MTP has to make double the number of network trips due to the EMS acting as message intermediary in the roundtrip.

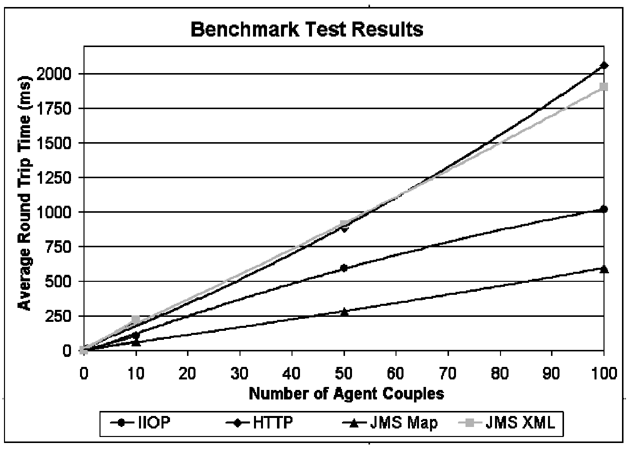

Figure 3. Benchmark Test Results

4.3. Benefits of JMS-MTP

The main advantage the JMS-MTP has over the current MTPs is guaranteed reliable messaging and flexibility. The JMS-MTP offers support for the redelivery of failed messages and message queuing for when a platform is offline or temporarily out of contact (network/service outage): - an agent platform is no longer required to be available for a message to be sent to it. The JMS-MTP is also more proficient in coping with high-volume traffic bursts, allowing for an agent platform to process messages at its own pace, buffering messages until it is ready to process them. The JMS-MTP also offers multiple message formats to maximise for performance or portability.

Through the use of the JMS-MTP, you are able to leverage high quality distributed infrastructure services written by domain leaders within the messaging middleware industry. When constructing large-scale systems it is vital to utilise an enterprise-level EMS/MOM implementation. Enterprise-level messaging platforms normally have built-in support for service federation, load-balancing and clustering, these services allow for massive-scalability (some commercial implementations support tens of thousands of clients with tens of thousands of operations per second) [15].

The JMS-MTP is a flexible solution for interconnecting agent platforms as it leverages the benefits of EMS/MOM in the areas of decoupling, scalability, system availability, and cohesiveness. This reduces the effort required to add and remove participants from an interconnected group of agent platforms.

The JMS-MTP allows for a simple and flexible integration of agent-based systems into heterogeneous enterprise systems. It provides a straightforward mechanism to easily integrate agent-based systems with traditional, non agent-orientated, based software solutions.

The JMS-MTP has been used to integrate a JADE agent system with a Java 2 Enterprise Edition (J2EE) based e-business system.

5. Future plans

The next stage of our research is focusing on the integration of the Publish/Subscribe messaging model, found in EMS/MOM, into the JADE agent platform. Along with a version 2.0 release of the JMS-MTP we also plan to develop the MTP into a Java Agent Services (JAS) [16] transport service, this would allow for the MTP to be easily used within other agent platforms. The definition of services, tools and techniques to further enable a streamlined integration of agents and agent-based systems into Enterprise Application Environments (EAE) is also being investigated.

6. Conclusions

This work forms the necessary next step in integrating agent-based systems into enterprise applications. This paper describes work-in-progress on agent-based messaging in the V-LAB project at the Department of Information Technology, National University of Ireland, Galway, Ireland.

Agent researchers have observed that one of the key obstacles to the widespread deployment of agent technology is the relative immaturity of agent technology with regard to scalability, federation, persistence, transactions, security, deployment lifecycle, management, and integration with legacy systems [6]. We believe that the current messaging capability of agent technology is also an obstacle to deployment in enterprise systems. To rectify this we have developed a Message Transport Protocol (MTP) for the Java Agent DEvelopment (JADE) platform that uses an Enterprise-Messaging Service (EMS), via the Java Messaging Service API, to transport *inter*-platform agent messages.

The JMS-MTP opens the world of EMSs to JADE; these services are used to build highly reliable, scalable and flexible distributed applications. EMSs provide a host of powerful advantages over conventional distributed computing models, they encourage loose coupling and provide guaranteed reliable message delivery.

The JMS-MTP for JADE offers a number of advantages to the developers of agent-based systems. These benefits include the ability to easily communicate and integrate agent-based systems with non agent-orientated systems. The use of JMS has removed vendor lock-in to a proprietary EMS implementation; this allows an agent platform to use the most relevant EMS for its requirements.

We believe the JMS-MTP moves JADE and agent platforms a step closer to streamlined flexible integration with business environments and full-scale participation in the digital enterprise.

7. Acknowledgement

The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

8. References

[1] FIPA, "FIPA Agent Message Transport Service Specification," 2002.
[2] FIPA, "FIPA Agent Message Transport Protocol for HTTP Specification," 2002.
[3] FIPA, "FIPA Agent Message Transport Protocol for WAP Specification," 2002.
[4] FIPA, "FIPA Agent Message Transport Protocol for IIOP Specification," 2002.
[5] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE - A FIPA-compliant agent framework," presented at PAAM, London, 1999.
[6] D. Cowan and M. Griss, "Making Software Agent Technology Available to Enterprise Applications," *HP Labs Technical Reports*, 2002.
[7] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, "JADE Administrator's Guide," 2002.
[8] "Orbacus ORB MTP", http://sharon.cselt.it/projects/jade/
[9] "HTTP MTP", http://liawww.epfl.ch/
[10] Sun Microsystems, "Java Message Service: Specification," 2001.
[11] FIPA, "FIPA Agent Message Transport Envelope Representation in XML Specification," 2002.
[12] JBoss Group - JBossMQ", http://www.jboss.org
[13] Sonic Software - Sonic MQ", http://www.sonicmq.com
[14] E.Cortese, F.Quarta, and G.Vitaglione, "Scalability and Performance of JADE Message Transport System," presented at AAMAS Workshop on AgentCities, Bologna, 2002.
[15] MSF Group, "Deployment Strategies focusing on Massive Scalability," 2003.
[16] JCP, "Java Agent Services API Specification", http://www.java-agent.org