Response to the FIPA Call for Technology on "Agents in Ad-hoc Environments"

1. Part I – Overview

1.1 Contact Information

Contact: Jamie Lawrence
Title: Research Fellow
Email: jamiel@mle.media.mit.edu
Address: Media Lab Europe
Sugar House Lane, Bellevue, Dublin 8, Ireland.
Telephone: +353 1 474 2868

1.2 Overview of the Proposal

The proposal presented in this document is derived from the work currently underway at Media Lab Europe to enable an existing, open-source, FIPA-compliant agent platform with the ability to operate in an ad-hoc environment. This mechanism provides facilities for discovering instances of an agent platform using current service discovery techniques whilst recognizing the serious resource constraints often found in mobile ad-hoc networks.

Four main modifications to the current FIPA specifications are proposed in order to support an ad-hoc environment: leased directory entries, a notification mechanism for directory changes, the removal of the Directory Facilitator (DF) and Agent Management System (AMS) as mandatory components, and the addition of a Discovery Agent (DA) to handle platform discovery and peer-to-peer agent discovery.

1.3 Completeness of the Proposal

This proposal is *untested* and only *partially complete*; it would require further work to be considered as final specification. In particular, the proposal is at a high level and describes the interactions between agents and a service discovery layer but does not specify the ontologies or service discovery protocols to be used.

The following issues would need to be addressed:

- Define an ontology for the Discovery Agent
- Modify the existing AMS and DF ontologies to support the leasing operations
- Specification for the representation of a FIPA agent platform in one of more concrete service discovery protocols. This would be similar to the way the concrete MTP protocols are specified.

2. Part II – Proposed Solution

2.1 Introduction

Our solution is motivated by the belief that the transient connections that exist in an ad-hoc network will enable innovative applications. By allowing an application to become aware of other agents passing by in the street, we can envisage applications emerging from the (possibly complex) communication between agents. These applications include networks of autonomous sensors, capable of analysing and processing data "in the field" and the sharing of information between people: file sharing, news filtering and even the construction and visualisation of networks of personal stories.

As such, the solution presented here is particularly focused on resource-limited nodes in mobile ad-hoc networks and the peer-to-peer interactions between them. However, we also describe a means to move from strictly P2P discovery to a version of the existing FIPA directory-based discovery mechanism. This work is largely focused on the practical aspects of deploying agents in ad-hoc networks, particularly the modifications to an existing platform (JADE-LEAP) and demonstration of emergent agent applications. Whilst we are striving to maintain compatibility with the FIPA specifications and JADE-LEAP, we also have considerable freedom in our design decisions – a privilege not shared by FIPA. Therefore, the solution presented here has been modified slightly to provide a better fit with current FIPA specifications and neglects certain details of our implementation.

2.2 Assumptions

- A Service Discovery protocol will be available to discover platform instances.
- A suitable routing protocol will be available for message transport. This should provide route discovery (either proactive or reactive) and may also allow multi-hop routing.
- Ad-hoc networks will consist of resource-constrained nodes. Each node will typically host only a single application agent, 2-3 at the most.

The platform discovery mechanism may be tightly coupled with the transport medium (such as the Service Discovery Protocol in Bluetooth) or a generic system such as JXTA[1] or Jini[2]. Routing algorithms are expected to handle the creation and maintenance of routes between platforms to avoid the inefficient message routing done at the level of the agent platform. Active routing protocols (such as OLSR[3], ZRP[4]) or information gathered directly from the physical layer may be used to aid the platform discovery process.

In contrast to existing agent platforms which are required to host many agents, we assume that a constrained device will host only a few application agents (typically just one) and therefore less emphasis is placed on common services (such as directories). However, the concept of a platform is still valid as it allows the abstraction of common functionality from the agent code and compatibility with larger environments.

2.3 Current FIPA Limitations

FIPA has specified two mandatory components of an agent platform: the directory facilitator (DF) and agent management system (AMS) that act as yellow and white page directories, respectively. These directories are necessary to support multiple agents on a single platform and efficient agent discovery but they also represent large, resource-consuming components, not suitable for deployment on embedded devices. It is clear then that both of these components must be removed from small ad-hoc nodes. However, since directories improve the scalability of the network, our modified platform will host directory services if the device is capable (in terms of memory, processor and network resources) and when the surrounding environment makes it necessary to do so (i.e. there exists a high density of small, unregistered nodes). See Section 2.6 for a further discussion on the removal of the AMS.

The current AMS and DF directories contain no mechanisms to actively heal themselves if a client fails to deregister before being disconnected from the network or moving out of range. This will quickly lead to inconsistent directories in an environment of frequent changes to the network topology. The Jini technology has popularised a leasing approach whereby directory entries are leased to individual services for a specified period. It is the responsibility of the service to renew the lease before it expires; when this occurs the entry is removed from the directory. A similar mechanism allows our modified DF and AMS to "self-heal" within the specified leasing period if an agent disappears from the network (or indeed becomes too overloaded to maintain the lease).

The typical request-response interaction requires the agents to actively poll a directory to receive new directory entries. In a dynamic environment, a more efficient publish-subscribe method would allow agents to subscribe to notifications when a new entry is added to the directory.

In summary, the proposed modifications avoid the restrictions imposed by the current FIPA standards by:

- adding a Discovery Agent (DA) to handle peer-to-peer platform and agent discovery.
- removing the DF and AMS as mandatory components of a platform, but allowing their activation should a device be capable and an environment require them.
- leasing directory entries within the DF and AMS
- providing a notification mechanism to allow the propagation of directory changes

2.4 Basic Discovery Events

The discovery process can be broken down into a number of individual events, each described below and shown in Figure 1. Within these descriptions, the term "fragment" is used to refer to a self-contained instance which is *not* hosting an AMS or DF. These fragments can form "compounds" by registering their agents with a platform (i.e., a FIPA-compliant entity with an AMS and DF). A compound is simply an abstract concept used to group together the fragments registered with a common platform and has no physical or virtual representation.

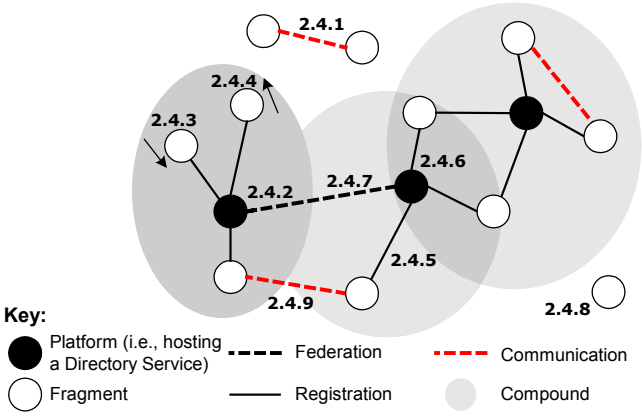

Figure 1: Basic Discovery Events

2.4.1 Direct Communication between Fragments

When two fragments meet in isolation, a form of peer-to-peer discovery must take place in order to allow communication between them. As the two devices discover each other, the discovery protocol notifies each discovery agent and they exchange their platform descriptions and the descriptions of *all* the agents each fragment is hosting. In addition, each fragment internally notifies each hosted agent of the newly discovered agents in the other fragment. The individual agents are responsible for examining the agent descriptions of the newly discovered agents and deciding if communication is possible, desirable or necessary.

This scenario is limited in scalability and will only support a few fragments.

2.4.2 Activation of Directory Services

One or more directory services (typically an AMS and DF) will be activated within a fragment according to a pre-defined strategy. On a constrained device (such as a mobile phone), this strategy may be simply "never host a directory". On a more capable device (that perhaps forms part of a backbone), the strategy would be "always host a directory", mimicking the current FIPA functionality. A wealth of strategies exists between these two extremes.

A possible strategy would involve monitoring metrics such as the number of discovery requests made and the number of local fragments not registered in a directory. If either of these measures crosses a specified threshold then a directory service is created (with a suitable random back-off time to ensure every fragment doesn't come to the same conclusion).

Once an AMS or DF is created, the discovery agent on that fragment will register the local agents with it (as explained more fully below). In addition, from this point on only the directory services will be advertised, and the individual agents must be discovered by first searching the directory service.

The deactivation of directory services will follow a similar pattern.

2.4.3 A Fragment Connects

When a fragment moves into range, it can detect another fragment hosting a directory service and subsequently register its hosted agents with this directory. Once the agents are registered with at least one (local or remote) directory, the fragment will only advertise that directory (not all of the hosted agents) unless specifically asked to do so. This mechanism forms a *compound* and allows smaller fragments to reduce their load during discovery by referring all search requests to the fragment with a directory service.

In cases where a fragment is discovered but its associated directory cannot be contacted (see Figure 2), the basic P2P discovery can take place between the two fragments (see section 2.4.1).

2.4.4 A Fragment Disconnects

When a fragment shuts down it may intentionally disconnect by deregistering its hosted agents from all directory services. However, more often a fragment is unexpectedly disconnected due to user intervention or network disruption. In these circumstances, the directory should self-heal as the leases on the directory listings begin to expire. The fragment will also recognise the disappearance of a directory through the same method (i.e., it will attempt to renew the lease only to find the directory not contactable). The fragment will continue to attempt discovery of other nearby fragments and the hosted agents will only be able to contact each other.

2.4.5 Registration of an Agent

Upon start-up, an agent registers its description with the local discovery agent. The discovery agent registers these descriptions with a directory service once one is discovered (covered in section 2.4.3).

2.4.6 Multiple Registrations of an Agent

An agent may be registered with multiple directory services at any one time, i.e., it may exist in more than one compound.

2.4.7 Federation of Directory Services

When a fragment hosting a directory service discovers another directory service, the two may federate together based upon some pre-defined strategy. This strategy may be simple: a timer to ensure the link is stable enough and to prevent spurious and short-lived federations.

2.4.8 Disconnected Fragments

A fragment may be completely disconnected from all other fragments (see section 2.4.4).

2.4.9 Communication between Fragments

Communication between agents happens as usual, with one caveat: due to the nature of wireless networks, it is possible for an agent to be discovered on a remote fragment that it is not possible to directly communicate with (see Figure 2). It seems intuitive that a route should exist between the two agents by using the directory to route the messages and in some cases the underlying transport protocols will provide multi-hop routing to fulfil this. However, it is worth noting that although a valid route might exist it does not guarantee that a routing protocol will have discovered it.

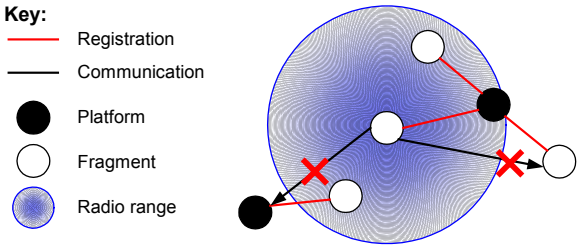

Figure 2: Wireless networks are not fully connected. An agent discovered through a directory may not be contactable due to the limitations of the wireless technology and the routing protocols in use.

2.5 Discovery Agent

The largest addition is the discovery agent (DA), which is responsible for advertising and discovering the presence of fragments, agents and directories in addition to controlling the activation of the local directory services. The DA is implemented as an agent with support for one or more discovery protocols and is executed when a discoverable container starts up.

The minimum information advertised by a discovery protocol is the agent identifier (AID) of the discovery agent. Further information may then be requested directly from the DA using standard agent communication. Reducing the amount of information shared by the discovery protocol allows the use of very simple protocols that cannot represent a whole agent description (such as SSDP[5]). The discovery agent will support the following functions: register, deregister, subscribe, unsubscribe, get-advertisement, get-all-agent-descriptions, and get-directories.

2.5.1 Internal Functions

1. *register, deregister*. This function allows an agent to register or remove its description with the discovery agent.
2. *subscribe, unsubscribe*. This allows a local agent to subscribe to the notifications that are broadcast when agents are discovered or disappear.
3. *get-directories*. This function returns all directories, both local and remote, where the discovery agent has registered the hosted agents.

2.5.2 External Functions

4. *get-advertisement*. This function enables a remote discovery agent to retrieve the advertisement for this fragment. If the agents on this fragment are registered with a directory then a reference to this directory is returned, otherwise the descriptions of each agent is sent back to the remote DA. In both cases, the platform description is also returned. This is not a replacement AMS/DF service as there are no methods for querying or searching – the descriptions for all currently registered agents are returned in response to a discovery request. Hence, this discovery mechanism is appropriate for only a small number of agents per fragment.

5. *get-all-agent-descriptions*. This performs exactly the same as the get-advertisement function when the agents are not registered with a directory service. This is used to force P2P discovery in the case where the initiator cannot access the directory service where the agents are registered (see section 2.4.9).

In response to a notification from the service discovery middleware, the local discovery agent will call the remote fragment's get-advertisement action. When one or more directories are returned two possible actions may occur. If the local agents are not registered with a directory, they will be registered with the directories returned. If a local directory exists then it will be federated with the returned directories (based on some strategy as previously mentioned).

2.6 Removal of the AMS and DF

The removal of the AMS and DF as mandatory entities allows for lower network, memory and processor costs, particularly in embedded environments where each fragment only supports a single agent. The costs of these services are related to the storage of the directory entries and the time required to process search requests. It is for these reasons that the discovery agent presented above does not perform any searching and only allows local registrations. On devices that are more capable the AMS and DF can be activated and utilised not only by local agents but also by those on nearby constrained devices.

The removal of the AMS has significant side effects that have not been mentioned previously; in addition to providing a white pages directory, the AMS is responsible for platform and agent lifecycle management. A more amenable modification to the design presented here would be to specify a "Lite-AMS" as a mandatory component of a fragment and integrate the DA functionality with that. This Lite-AMS would remain responsible for the agent lifecycle and platform resource but would lack the search and get-platform-description functions of the existing AMS. In addition, the Lite-AMS would only allow local agents to register with it. This removes the need for leasing Lite-AMS entries, as all agents will be locally hosted.

2.7 Compatibility Issues

2.7.1 Target Environment

The eventual target environment for these modifications will be smaller-than-phone embedded devices. Therefore, the platform should be able to comfortably operate with a single application agent in (much) less than 512k of memory.

2.7.2 FIPA Compatibility

With regards to FIPA compatibility, it is debatable whether this platform can comply with the current standards. On the surface, the removal of the AMS and DF as mandatory platform components fails to comply with both the FIPA Agent Management[6] and Abstract Architecture specifications[7]. However, the DA can be viewed as an inefficient directory service, which in response to a query performs no filtering and returns all registered entries. Viewed in this way, each DA fulfils the role of a directory and our fragments can therefore comply with the *abstract* notion of an agent system but not with the current Agent Management specification.

2.8 Conclusions

The design presented here will allow the deployment of agents in an ad-hoc network. A number of modifications have been proposed that will allow an agent platform to discover other platforms, thereby allowing the agents to begin interacting with each other without the existence of a fixed infrastructure. These modifications are specifically targeted to embedded devices and are independent of the discovery or routing protocols employed. We have already begun applying these modifications to an existing FIPA-compliant platform.

2.9 References

[1] Project JXTA. JXTA v1.0 Protocol Specification. http://www.jxta.org.

[2] Waldo, J., Arnold, K., and The Jini Team. The Jini Specifications. Addison-Wesley, 2000.

[3] Clausen, T., Jacquet, P., Laouiti, A., Minet, P., Muhlethaler, P., Qayyum, A., and Viennot, L. Optimized Link State Routing Protocol. http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-06.txt.

[4] Haas, Z. J. and Pearlman, M. R. ZRP - A Hybrid Framework for Routing in Ad Hoc Networks. Ad Hoc Networking, Perkins, C., 2001, 221-253.

[5] Yaron Goland, Ting Cai, Paul Leach, Ye Gu, and Shivaun Albright. Simple Service Discovery Protocol. Internet Engineering Task Force.

[6] Foundation for Intelligent Physical Agents. [FIPA00023] FIPA Agent Management Specification.

[7] Foundation for Intelligent Physical Agents. [FIPA00001] FIPA Abstract Architecture Specification.