LEAP for Ad-hoc Networks

Jamie Lawrence
mailto:jamiel@mle.media.mit.edu

Disclaimer
This document is a work-in-progress and is subject to change.

Abstract

This document is intended to provide an overview of the effort to enable the Lightweight Extensible Agent Platform (LEAP) to operate in ad-hoc networks. I discuss the motivations behind this project, the usage scenarios and finally the required modifications to LEAP. With this document I wish to inform the members of the LEAP, JADE, and FIPA communities about my research plans and solicit any feedback they may wish to offer.

Approach

The approach taken by this project is to integrate existing peer-to-peer (P2P), service discovery and ad-hoc networking technologies into the LEAP platform to provide a robust infrastructure for deploying agents within an ad-hoc network.

Ad-hoc networks are the subject of current research into future generation networks but like most new technologies the term is often misused and misunderstood.

> **ad hoc** adj
> 1: often improvised or impromptu; "an ad hoc committee meeting"
> 2: for or concerned with one specific purpose; "a coordinated policy instead of ad hoc decisions" adv : for one specific case; "they were appointed ad hoc"
> Source: *WordNet ® 1.6, © 1997 Princeton University*

Due to the current focus on ad-hoc *routing protocols* there has been a trend to classify ad-hoc *networks* as just those networks that require multiple hops to deliver a message. We do not limit our scope in this way. Neither do we require an ad-hoc network to operate using radio technology; infra-red and fixed wireline networks can exhibit the same ad-hoc properties and benefit from the results. This document uses the term “ad-hoc” to refer to the presence of a decentralised infrastructure, unpredictable nature and transient local interactions between multiple nodes.

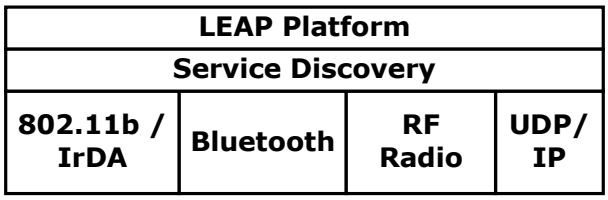

Figure 1: The "Big" Picture

Certain assumptions are made by this project:

- As shown in Figure 1, a Service Discovery middleware will be available to notify the agent platform of changes to its local environment. This mechanism may be tightly coupled with the transport protocol (such as the Service Discovery Protocol in Bluetooth or IP broadcasts) or a more generic system such as JXTA[10] or Jini[8]. Low-level active routing protocols (such as OLSR[11] or ZRP[12]) or information gathered from the physical layer may be used to aid the discovery process.
- Most suitable transport protocols will take care of all multi-hop routing and network-level navigation, such as NAT and firewall traversal. The majority of the research

into ad-hoc networks and P2P systems has been in this area and it is generally preferable to platform-based routing.

- Constrained devices will typically host only a single agent and therefore less emphasis needs to be placed on common services (such as directories) within the platform. 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.

Terminology[1]

See Figure 2: Basic Usage Scenarios for a visual indication of these terms.

Fragment

A fragment, for the purposes of this document, consists of a JADE-LEAP container hosting one or more agents. A fragment resides on a single device and may optionally host a directory service such as an Agent Management Service (AMS) and/or Directory Facilitator (DF).

Compound

A compound is a collection of fragments registered with a common directory service. Note that fragments may be registered with multiple directory services.

Federated

Connected so as to appear as a single entity to the user. For example, a search can take place across many federated directories that appear as a single directory to the user.

Federation

A federation is a collection of compounds with federated directory services.

Platform

There is no longer the notion of a static FIPA-compliant platform since any fragment may become FIPA compliant depending on its configuration and circumstances. Use of this term within this document typically refers to the code under development rather than a runtime entity.

Directory Service

A Directory Service is simply an AMS and/or DF. The term is used when no distinction between the AMS and DF is required.

Discovery Service

A new service that will be developed as part of this project and that is explained more fully below.

Neighbourhood and Locality

If any reference is made to locality it is in the context of the network being used; an IP network might use subnets, a wireless network might use signal strength or range, and multi-hop networks might use the hop count of a route.

FIPA-related Terms

DF Directory Facilitator is a yellow-pages directory service
AMS Agent Management System is a white-pages directory service
MTS Message Transport Service is the communication mechanism used by FIPA platforms

[1] Much of the terminology and inspiration for this work comes from FIPA [3], the LEAP Project [2,7] and the JADE platform [1].

Jamie Lawrence jamiel@mle.media.mit.edu

Motivations

The current implementation of LEAP, like that of its JADE ancestor, has a static structure consisting of one main container providing FIPA interoperability to one or more (possibly lightweight) containers. Although containers may join or leave at any time, the platform includes no mechanism to actively handle this and maintain a consistent directory. It is not possible to reassign the main container's functionality to another container at runtime or to start a lightweight container without the existence of a main container. The ability to run a lightweight container without depending on a main container or running an AMS and DF locally is important, and therefore I would like to address this shortcoming in this project.

Crucially, the LEAP platform includes no mechanism for dynamically discovering containers at runtime. This will limit its deployment in an ad-hoc network and is therefore the primary focus of this project.

Requirements

- Removal of AMS, DF and Main container as mandatory requirements for hosting agents
- Discovery of fragments
- Dynamic formation of fragments into compounds and associated strategies
- Dynamic creation of directory services within fragments and associated strategies
- Leased directory entries to ensure consistency over time
- Active updates to the directory services as agents join and leave the network
- Directory subscription mechanism to enable fragments to receive notifications
- JADE API compatibility from the *agent developers* viewpoint

Basic Usage Scenarios

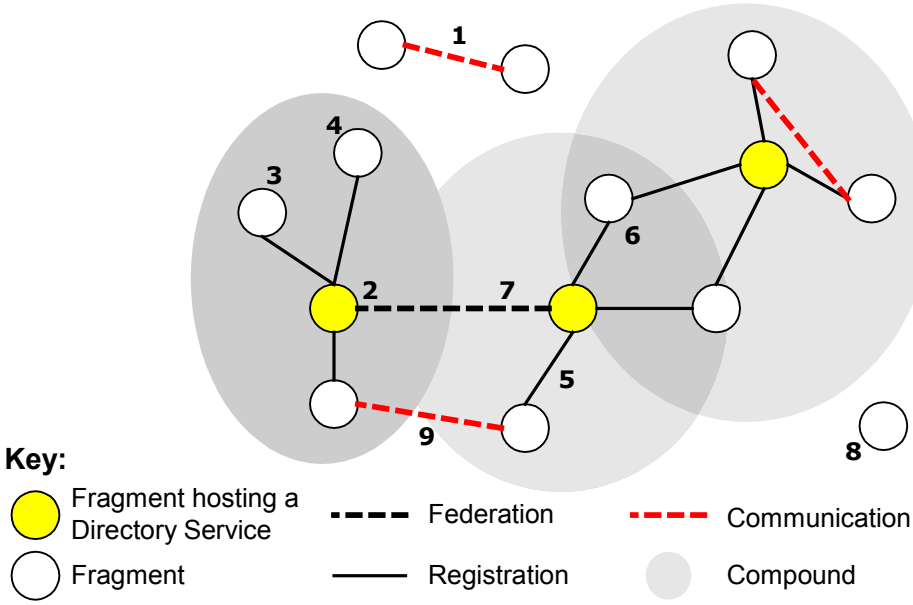

Figure 2: Basic Usage Scenarios

1. Direct Communication between Fragments

This scenario describes direct communication between two fragments using some peer-to-peer discovery technique rather than any form of directory service. As the two devices discover each other their platform descriptions and the descriptions of *all* the agents each fragment is hosting are exchanged. In addition, each fragment internally notifies each hosted agent of the newly discovered agents in the other fragment.

This scenario is limited in scalability and would only be used between a few fragments.

Jamie Lawrence jamiel@mle.media.mit.edu

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 JADE-LEAP functionality. A wealth of strategies exists between these two extremes.

One of these strategies might involve monitoring the number of P2P discovery requests being 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 may be created (with a suitable random back-off time to ensure every fragment doesn't come to the same conclusion).

Once a directory service has been created the discovery service will register all of 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 would follow a similar pattern.

3. A Fragment Connects

When a fragment has become discoverable (i.e. it has been created or moved into range) it will detect the presence of a directory service belonging to another fragment and subsequently register all its hosted agents with this directory. Once the agents are registered with at least one (local or remote) directory service the fragment will only advertise that directory service (not all of the hosted agents) unless specifically asked to do so. In this way individual fragments can reduce their load during discovery by referring all search requests to a fragment with a directory service. In cases where a fragment is discovered but its associated directory cannot be contacted, the basic P2P discovery can take place between the two fragments (see Scenario 1).

4. A Fragment Disconnects

If a fragment is being shut down then it may perform intentional disconnection by deregistering its hosted agents from all directory services. However, more often the fragment will be unexpectedly disconnected due to user intervention or network disruption. In these circumstances the fragment and directory services might be notified of this by the service discovery middleware and take the appropriate clean-up action. In any case, as all directory listings are leased, the directory services will self-heal once those leases expire (Jini-style[8]). The fragment will also recognise the disappearance of a directory through the same method. The fragment then will again begin using P2P discovery / advertisement techniques.

5. Registration of an Agent

An agent is registered with a directory service once the fragment has discovered one. This is covered in Scenario 3.

6. Multiple Registrations of an Agent

An agent may be registered with multiple directory services at any one time.

7. Federation of Directory Services

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

Jamie Lawrence jamiel@mle.media.mit.edu

8. Disconnected Fragments

A fragment may be completely disconnected from all other fragments. In this case, the fragment will continue to attempt discovery of other nearby fragments and the hosted agents will only be able to contact each other.

9. Communication between Fragments

Communication between agents happens as usual, with one caveat: due to the nature of wireless networks and the federation of directories, it is possible for an agent to be discovered on a remote fragment which you are unable to directly communicate with. For this reason, all agents registered with a remote directory service must register two transport addresses: that of the fragment they're hosted on and the address of the the directory service through which they were discovered. In this way messages will first be sent directly to the agent's fragment and if this proves unreachable the message will be routed via the directory service. In some cases the underlying transport protocols will provide multi-hop routing to replace this mechanism.

Modifications of the LEAP Platform required to support Ad-Hoc Networks

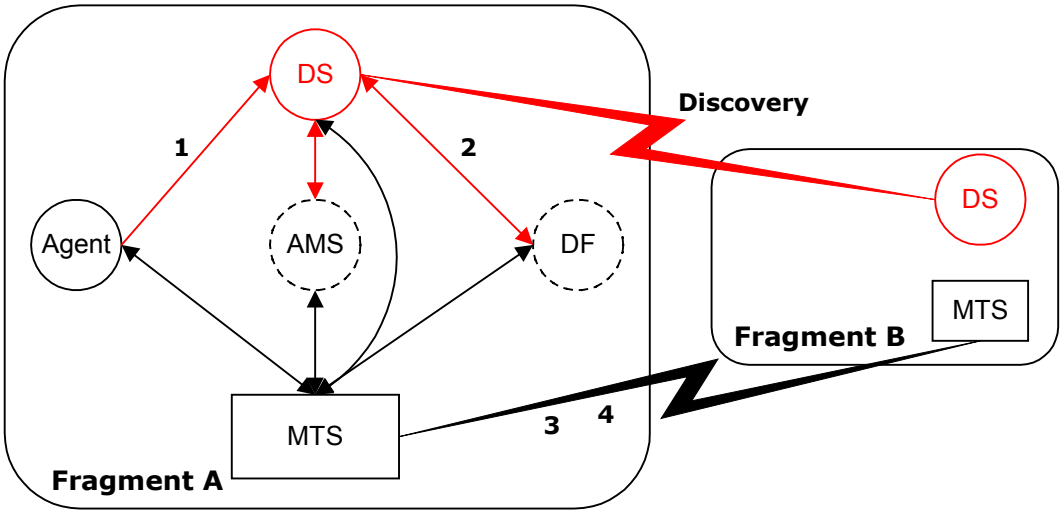

Figure 3: Overview of the modifications to LEAP. Red indicates new discovery-related communications or components. Dotted lines indicate an optional component.

Several modifications must be made to enable the LEAP platform within an ad-hoc environment (see Figure 3).

Discovery Service

The largest modification is the addition of a discovery service[2] (DS) that is responsible for advertising and discovering the presence of fragments, agents and directories in addition to controlling the activation of the directory services. This discovery service is implemented as an agent with support for one or more discovery mechanisms[3].

The minimum information advertised by a concrete discovery protocol is the name and type of the fragment and the AID of its discovery service. Further information may then be

[2] See "Appendix B: Compatibility Issues" for a discussion on whether the functionality of the discovery service could be merged with the AMS.
[3] See "Appendix A: Modelling the Discovery Service as a Platform Component" for a discussion on whether the discovery service should be implemented as an agent or a software component.

Jamie Lawrence jamiel@mle.media.mit.edu

requested directly from the discovery service using standard agent communication[4]. The discovery service will support the following functions: *register*, *deregister*, *subscribe*, *unsubscribe*, *get-advertisement*, *get-agents.*

(1) register, deregister. This function allows an agent to register or remove its description with the discovery service. This function is only available to local agents.

(2) subscribe, unsubscribe. This allows a local agent to subscribe to the notifications that are broadcast when agents are discovered or disappear.

(3) get-advertisement. This function enables a remote discovery service to retrieve the advertisement for the (local) fragment. If the agents on this fragment are not registered with a directory then a description of each agent is returned to the remote fragment. Otherwise, a reference to the directories (local or remote) is returned. In both cases the platform description is also returned.

Note that 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.

(4) 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 scenario 3, page 4.

In response to a notification from the service discovery middleware, the local discovery service 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).

Directory Services (AMS, 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 service presented above does not perform any searching and only allows local registrations. On more capable devices the AMS and DF can be activated and utilised not only by local agents but also those on nearby constrained devices.

In contrast with the existing DF/AMS all directory entries are leased and must be renewed prior to expiration in a similar manner to Jini[8]. Additional functions *subscribe* and *unsubscribe* are also required to allow the propagation of directory changes to individual agents.

As has been implied the AMS will also provide DF-style federation as the current method of FIPA dynamic registration can become unmanageable[9].

Agent

The core Agent class requires only a few modifications which includes default registration with the local discovery service rather than the AMS and modifications to the DF and AMS communicators to hide the possible absence of these directories.

[4] This two-phase mechanism allows for service discovery protocols which are not flexible enough to represent complex agent descriptions. With a powerful service discovery language these two steps may be combined.

Jamie Lawrence jamiel@mle.media.mit.edu

Target Environment

The eventual target environment for these modifications will be smaller-than-phone embedded devices. However, initial development will be performed on Windows 2000 using a fixed IP network with an intermediate stage using PersonalJava on Compaq iPAQs. Wireless technology may be in the form of IrDA, Bluetooth, 802.11a/b or a simple RF radio, but has yet to be decided. The platform should be able to comfortably operate with a single agent in (much) less than 512k of memory.

Timescale

The initial development of this platform (including the core features) is estimated to take 6 months culminating in a demonstration in September 2002. Subsequent development may then be required.

References

[1] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa, "JADE - A FIPA-compliant Agent Framework" PAAM '99. See http://jade.cselt.it/papers/PAAM.pdf

[2] Federico Bergenti and Agostino Poggi, "LEAP: A FIPA Platform for Handheld and Mobile Devices" ATAL 2001. See http://leap.crm-paris.com/public/docs/ATAL2001.pdf

[3] Foundation for Intelligent Physical Agents, "[f-out-00105] FIPA TC Ad-hoc First Call For Technology"

[4] Foundation for Intelligent Physical Agents, "[FIPA00023] FIPA Agent Management Specification"

[5] Foundation for Intelligent Physical Agents, "[FIPA00094] FIPA Abstract Architecture Specification (Refinements)"

[6] Foundation for Intelligent Physical Agents, "FIPA 97 Specification Part 1 Agent Management" 10-10-1997. Full article at http://www.fipa.org/specs/fipa00019/OC00019A.pdf

[7] Giovanni Adorni, Federico Bergenti, Agostino Poggi, and Giovanni Rimassa, "Enabling FIPA agents on small devices" CIA '01. See http://leap.crm-paris.com/public/docs/CIA%202001.pdf

[8] Jim Waldo, Ken Arnold, and The Jini Team, "The Jini Specifications" 15-12-2000

[9] Michael Berger and Michael Watzke, "FIPA Ad Hoc Proposal Draft" 2002

[10] Project JXTA, "JXTA v1.0 Protocol Specification" See http://www.jxta.org

[11] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot, "Optimized Link State Routing Protocol" 31-10-2001. See http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-05.txt

[12] Zygmunt J.Haas and Marc R.Pearlman, "ZRP - A Hybrid Framework for Routing in Ad Hoc Networks" in Ad Hoc Networking, Charles Perkins (pages 221-253) 2001

Appendix A: Modelling the Discovery Service as a Platform Component

Originally the Discovery Service was modelled as a platform component in the same way as the MTS is currently implemented (i.e. not as an agent). After considerable thought the Discovery Service was described within this document as an agent acting within the platform in a similar way to the DF or AMS. However, the question is still open: *Should the Discovery Service be represented as an agent or a service?*

In either case it doesn't change the underlying purpose and functionality of the service, only the implementation and integration with the LEAP platform.

The Discovery Service as an Agent

The Discovery Service can be modelled as an agent with a number of behaviours to handle multiple discovery protocols. This would be a more natural model for an entity which can be described as having "strategies". In addition, it allows discovery to take a two-phase approach. The first phase advertises the existence of a fragment, its name, type and the address of the discovery service using some unspecified discovery methods. The second phase performs discovery of agents and directories from the discovery service and would be conducted using ACL via the existing MTS. This two-phase approach may be useful when the service discovery protocol supports a limited message format.

A possible advantage is that the implications on the LEAP source code are minimised if all discovery related activities are be contained within an agent and some associated behaviours.

It also seems easier to specify (from a FIPA perspective) the discovery-related interactions as ACL messages between agents rather than as interactions with a software component.

The Discovery Service as a Platform Component

The FIPA Abstract Architecture[5] acknowledges that a service may be implemented either as an agent using ACL messages or as a software component with a programmatic interface. In the past, the ACC within a FIPA platform was modelled as an agent [6] but this requirement has now been relaxed due to some implementation and theoretical issues. In particular, the ACC (as an agent) would have had the ability to refuse to send a message which is not considered a desirable property of this service[5].

Discovery can be viewed "as part of the agent environment" in a similar way to message transport and therefore shouldn't be elevated to the level of an agent (just as the ACC is no longer represented as an agent).

Comments

Personally I am neutral regarding this issue but it does appear that modelling the Discovery Service as an (constrained) agent has a number of advantages including the use of agent communication for discovery messages.

[5] If anyone who was around at the time of these discussions wants to elaborate or correct me, please contact me!

Jamie Lawrence jamiel@mle.media.mit.edu

Appendix B: Compatibility Issues

FIPA Compatibility

The goal of this project is to create an agent platform with which we can study the emergent properties of agent communities in a dynamic environment. We are therefore taking a pragmatic approach this project although it is intended that the results will be applicable to the standardisation efforts of the FIPA technical committee for Ad-hoc Networks (FIPA TC Ad-hoc).

With regards to FIPA compatibility, it is debatable whether this platform can comply with the FIPA Abstract Architecture specification[5]. On the surface, the removal of the AMS and DF as permanent platform components would fail to comply with the mandatory requirement for a directory service set out in the FIPA Abstract Architecture. However, the discovery service can be viewed as an inefficient directory service that in response to a query performs no filtering and returns all registered entries. Viewed in this way each fragment can comply with the abstract notion of an agent system but not with the current Agent Management specification[4].

One modification to the design presented here would be to leave the AMS as a mandatory component of a fragment [9] and integrate the functionality of the discovery service with that of the current AMS. This would make the design less radical and more amenable to the FIPA community without seriously impacting the flexibility of the platform although it has the affect of overburdening the AMS with responsibilities.

For practical reasons, the implementation of a fragment in a constrained environment may not be able to use a FIPA compliant transport protocol and therefore the current internal message transport of LEAP will be used. This in no way affects the quality of this work as a basis for future standards.

JADE-LEAP Compatibility

It is the aim of this work to maintain the existing JADE/LEAP API's, as seen from the perspective of the agent developer. Internal API's will be altered but where possible they will retain their existing functionality.

The JADE concepts of a *Main Container* and multiple *Containers* will no longer apply in the same sense – each device will host one or more fragments that will dynamically run directory services as required (or not at all). For this reason we do not use the terms "main container" or "container" as they imply some static assignment of status.

Jamie Lawrence jamiel@mle.media.mit.edu