Efficient Networking for Pervasive eHealth Applications*

Heikki Helin[1], Tim van Pelt[2], Michael Schumacher[2], Ahti Syreeni[1]

[1]TeliaSonera Finland Oyj
P.O.Box 970, FIN-00051 Sonera, Finland
{heikki.j.helin, ahti.syreeni}@teliasonera.com

[2]Swiss Federal Institute of Technology Lausanne (EPFL)
IN (Ecublens), CH-1015 Lausanne, Switzerland
{tim.vanpelt, michael.schumacher}@epfl.ch

Abstract: This paper presents the networking architecture developed in the CASCOM research project. This architecture provides an efficient and reliable communication support and service discovery for pervasive intelligent P2P business applications. We give an overview how wireless environments and resource-poor devices are taken into account in our architecture. For that, we have designed and implemented a new agent platform for resource-poor devices, which is a modified version of JADE/LEAP. Agents are used to coordinate (semantic) web services in the health care domain. To support the discovery of such web services descriptions, we shortly describe a distributed directory system called WSDir. It is based on federation of repositories of web service descriptions that are managed with pre-defined policies. Our architecture is being evaluated using a sample ad-hoc emergency health care assistance application scenario.

1 Introduction

In this paper, we present the networking architecture developed in the CASCOM research project†. The main objective of CASCOM is to implement, validate, and trial value-added supportive infrastructure for business application services for mobile workers and users across mobile and fixed networks. The driving vision of CASCOM is that ubiquitous business application services are flexibly coordinated and pervasively provided to the mobile worker/user by intelligent agents in the dynamically changing contexts of open, large-scale, and pervasive environments. The application domain is pervasive eHealth.

To achieve these goals, a generic CASCOM architecture has been defined to combine intelligent agent technology, semantic Web services, peer-to-peer, and mobile computing for intelligent peer-to-peer mobile service environments. For this purpose, a layered

* This work has been supported in part by the European Commission under the project grant FP6-IST-511632-CASCOM.

† www.ist-cascom.org

approach has been followed. The *Networking Layer* is a generic, secure, and open intelligent P2P (IP2P) network infrastructure taking into account varying Quality of Service (QoS) of wireless communication paths, limitations of resource-constrained mobile devices, service discovery, and contextual variability of nomadic environments. The IP2P network architecture can be used in several different network configurations including WLAN and several different kinds of WWAN networks (e.g., GPRS and UMTS). The *Service Coordination Layer*, situated above the networking layer, provides flexible semantic web service discovery, matchmaking, composition and execution functionalities. Orthogonal to both layers, the *Context Subsystem* is in charge of acquiring, storing and providing context information. *Security and Privacy Subsystem*, also orthogonal to the Networking and Service Coordination Layer, is responsible for ensuring security and privacy of information throughout the different components of the CASCOM infrastructure. Applications make use of the CASCOM generic architecture essentially through the Service Coordination layer. The Networking layer, as well as Context and Security and Privacy Subsystems, need to access the underlying networks.

In this paper, we present the networking layer of the generic CASCOM architecture. Especially, the networking layer provides (1) an agent execution environment for resource-constrained mobile devices, (2) efficient and reliable agent message transport communication over wireless (and wireline) communication paths independent of the access technology, (3) service discovery in IP2P environments, and (4) context information monitoring of network-related context information (e.g., QoS of a network, network availability, etc.). Further, necessary security and privacy requirements must be met in order to deploy the CASCOM architecture in real-world business applications. However, in this paper we discuss neither context information nor security and privacy issues.

The rest of this paper is organized as follows. In Section 2, we discuss a real-world use case scenario to illustrate the kind of domains that the proposed architecture aims at. Section 3 describes the agent platform designed and implemented for resource-poor mobile devices. In Section 4, we describe the CASCOM communication architecture that takes into account the possibilities and limitations of wireless communication paths. Section 5 presents our service discovery architecture. Finally, Section 6 concludes the paper.

2 Motivation

The business application scenario that we use in CASCOM is taken from the health care business domain, although our architecture does not contain any features that are specific to this field. The health care domain is to be one of the most viable and growing application field for intelligent mobile service coordination, not just for the technical requirements that it imposes, but also for its huge economic and social relevance. Benefits of using ICT in health care domain have the potential to significantly impact the quality, accessibility, and cost of health care. Due to the characteristics of the CASCOM health care scenarios, we believe that the application of the CASCOM technology has the potential to contribute to all three categories mentioned above

Consider the following sample scenario:

> *Rosi, a tourist from Finland, is visiting Portugal, and she suddenly fells ill. Her personal CASCOM agent installed on her PDA quickly finds out the contact information of a local emergency centre, so that Rosi gives the noticeable symptoms of her sickness, her location and some general information. The local representative orders her to go to the nearest hospital immediately.*
>
> *After Rosi's call, her personal agent contacts the Emergency Medical Assistance (EMA) CASCOM agent in Finland and requests to send Rosi's medical history to the Portuguese hospital. During the first examinations by the physician, there are some doubts about the diagnosis and she wants to obtain a second opinion, so she searches for a second opinion service using her CASCOM agent. After having found a specialized cardiologist, the agent provides Rosi's symptoms and medical records. The cardiologist asks the physician to provide more precise information about a particular symptom (e.g., if the body area affected by the pain has been operated in the past). Finally the cardiologist provides her opinion, but it is not sufficiently clear to the physician in a certain part, so she asks for a further explanation.*
>
> *The local physician, the patient, and EMA jointly take the decision that Rosi should be transferred back to Finland as soon as possible. Rosi's personal agent automatically finds out possible flight arrangements and informs all people that are involved during the travel (e.g., possible doctors and escorts). The agent also makes all the arrangements with a Finnish hospital. Back in Finland, Rosi is treated at a hospital in Helsinki.*

Mobile communication is essential to this scenario: Rosi, a mobile user getting sick in a foreign country may get in touch with various relevant people and services for help either personally or by using her personal CASCOM agent residing on her personal mobile device. For mobile workers such as EMA's Finnish representative in the above scenario, the CASCOM architecture enables ad hoc and timely communication and access to relevant information in any place, any time. Further, agents play an essential role in helping mobile workers performing their business tasks by finding and composing relevant (semantic Web) services on demand. Mobile workers gain more time for tasks that only humans can do, but a necessary support from underlying architecture is needed in order to deploy reliable applications in mobile/wireless environments. In what follows, we describe how the CASCOM architecture takes these issues into account. More detailed descriptions of our use case scenarios can be found in [CAS05].

3 Agent Platform for Mobile Devices

Agents in the CASCOM architecture need an agent platform that is usable in resource-constrained devices. The CASCOM agent platform is based on JADE/LEAP [BPBC01], which already provides the basic agent platform functionality needed by the CASCOM architecture. Although there are other agent platform designed for mobile devices (e.g., A-Globe [SPP04] and MicroFIPA-OS [LTL02]), the JADE/LEAP is to our best knowledge the only one which fully supports FIPA standards and is actively maintained.

JADE [BCPR03] is a distributed FIPA-compliant agent platform that allows agents to be executed on desktop computers in J2SE environment. When an extra component, LEAP add-on for JADE, is added to JADE, it can be also compiled for small devices, for example, PDAs and mobile phones supporting CLDC 1.0 and MIDP 2.0. This combination is called JADE-LEAP platform.

The CASCOM agent platform is based on JADE/LEAP. JADE/LEAP, however, cannot be used in pure IP2P network configurations as it expects some infrastructure components (known as main containers running on a mediator machine; see Figure 1), of which existence cannot be assumed, for example, in pure P2P network architectures. Therefore, we have designed and implemented necessary changes to the JADE/LEAP so that it can be executed in resource-poor devices supporting the J2ME CLDC 1.1 and that it does not need any infrastructure components outside the local device. Figure 2 depicts the basic agent platform configuration in our infrastructure.

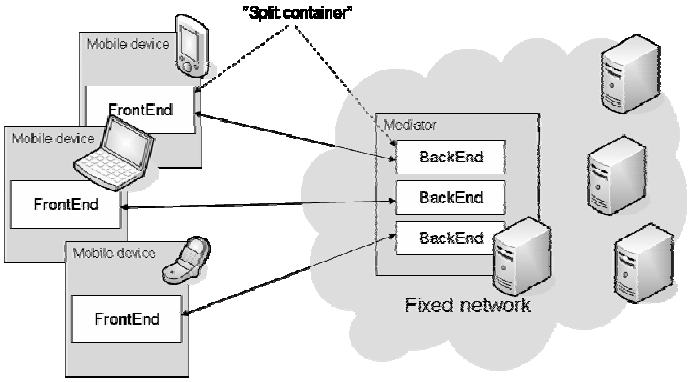

Figure 1: Split container model of LEAP

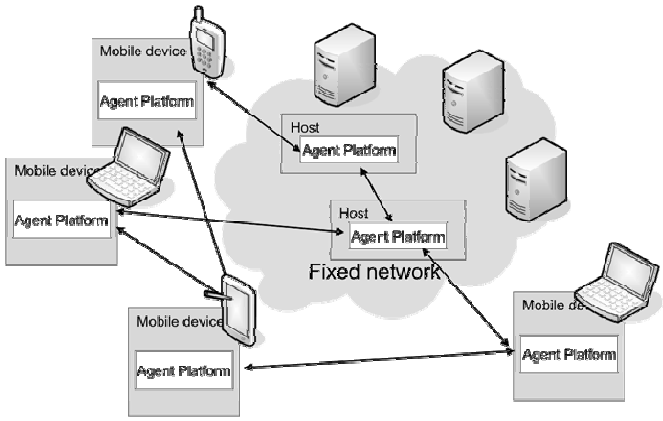

Figure 2: CASCOM agent platform configuration

The architecture of the CASCOM agent platform follows the design of the original JADE platform. The main components are depicted in Figure 3. The core services of JADE are loaded at startup. The main difference to the original JADE is that internal message protocol (IMTP) does not exist, thus there cannot be more than one container because the messaging between containers would require an IMTP implementation. However, the Figure 3 gives a simplified overview of the CASCOM agent platform main components, details are more complicated as there are still most of the JADE core classes left. Many of the core classes are needed to start the main container and cannot be removed unless making intensive changes to the current architecture of JADE.

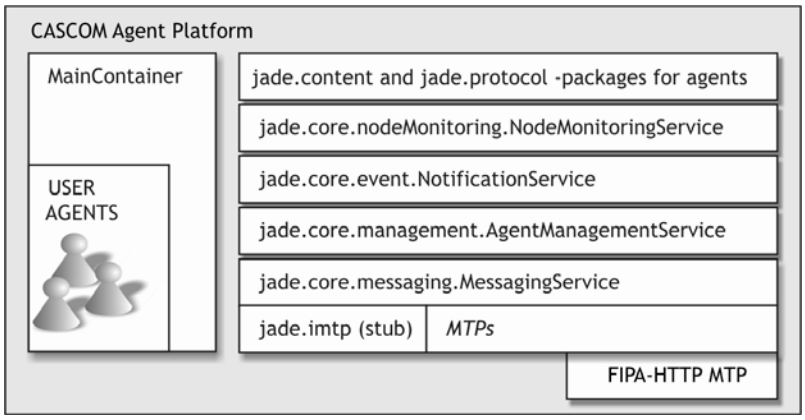

Figure 3: Overview of the CASCOM agent platform main components

4 Agent Communication in Wireless Networks

This section gives an overview of the CASCOM agent communication architecture. Here we will take a pragmatic view to agent communication. In particular, we neither consider why agents are communicating nor the semantics of messages. However, we assume that agents are communicating with one another and at least part of the communication path is implemented using wireless technology. The latter assumption is an additional requirement in the sense that many of the solutions provided are applicable also in environments where the whole communication path is implemented using wireline technologies. The agent communication in the CASCOM architecture is based on FIPA standards.

4.1 Communication over Wireless Communication Paths

Efficient agent message transport is an essential feature of the CASCOM networking architecture as wireless communication paths are used. Without this, usability of the CASCOM architecture would not be optimal. Efficient agent message transport includes appropriate agent message encoding (e.g., using efficient binary encoding instead of

more verbose string-based encoding) and a message transport protocol that can transfer messages reliably and efficiently.

Communication between agents can be depicted as layered model (see Figure 4). The transport and signaling protocol layer should provide an efficient and reliable data transport service. Usually this layer should be transparent to agents. Therefore, the agents are typically unable to optimize anything at this layer by themselves. Given this, we will not discuss these issues in more detailed here. An overview of transport protocols in wireless environments can be found in [MD00], as an example.

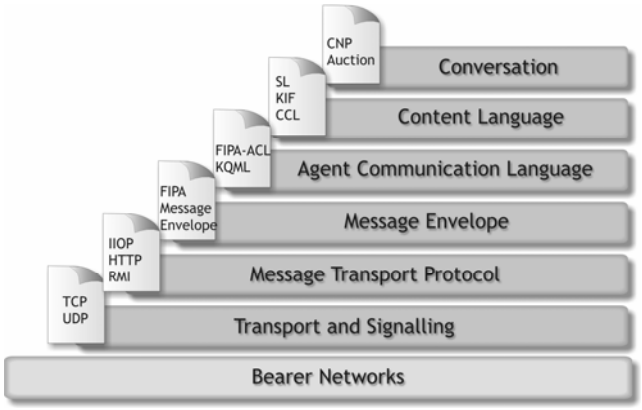

Figure 4: A layered model of agent communication

The message transport protocol (MTP) defines the structure of messages sent using a transport protocol. The MTP typically implicitly defines also the transport protocol, in which case the agents do not have to care which one is used. In the CASCOM architecture, we use HTTP as an MTP [FIP00b]. HTTP is not an optimal protocol for communication over wireless links, but is easy and quite light-weight to implement in resource-poor devices.

The message envelope, the agent communication language (ACL) and content language layers are treated as encoding layer. Although all these layers have their own encoding, the basic principles for optimizing them to the wireless world are the same.

The message envelope defines, among other parameters, how the message should be routed. In the CASCOM architecture the message envelope is encoded using bit-efficient envelope encoding defined by FIPA [FIP00c]. The encoding is similar to that of ACL and gives similar results as the ACL encoding (see below).

The agent communication language defines the syntax and the semantics of the messages agents are sending to each other. In the CASCOM architecture we use FIPA-ACL encoded using bit-efficient syntax. Here we concentrate on the bit-efficient encoding [FIP00a], which is especially suitable for wireless environments. In the bit-efficient ACL, there are two primary ways to reduce the transfer volume over the wireless link: data reduction and intelligent caching. First, ACL messages are encoded efficiently by

using one-octet codes for predefined message parameters and other common parts of messages. This is a significant improvement compared to a simple string-based coding, as it typically reduces extra overhead to half of the original. Furthermore, this improvement is easy to implement and faster to parse than the string-based coding—comparing bytes is typically much faster than comparing strings (see [He03] for details). Figure 5 gives an example of the effect of different ACL encodings. As shown in Figure 5, the bit-efficient ACL encoding is the most efficient—even compressed string-based representation produces larger output. Although the difference between bit-efficient and compressed string-based encoding is insignificant, there is another advantage of using the bit-efficient encoding; constructing and parsing bit-efficiently encoded messages is faster. The main reason for this is that should a message be compressed using a general-purpose compression algorithm, the message has to be parsed after the decompression in order to create an appropriate Java object. In the bit-efficient encoding, this phase is built-in into the parser. More detailed results of the bit-efficient ACL encoding can be found in [HL02, He03].

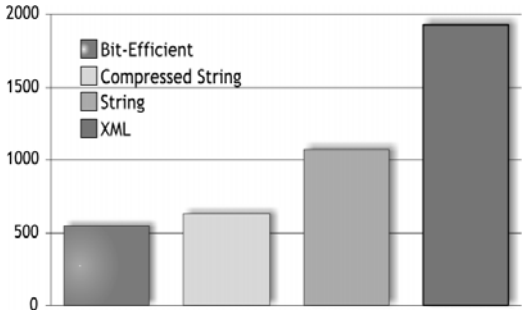

Figure 5: The output size in bytes using different ACL encodings

In the CASCOM communication architecture, the message content is expressed using FIPA-SL. This language does not have a bit-efficient syntax like the message envelope or ACL. Therefore, in the CASCOM communication architecture, the message content is compressed using a standard deflate algorithm. This gives good results if considering only the output size. However, for compressing and decompressing somewhat significant amount memory is needed, and thus it is not a suitable technique for cases where amount of memory is very limited. We are currently looking for less memory extensive algorithms to compress the message content.

4.2 Messaging Gateway

Due to unreliable wireless connections, possible firewalls and NAT, the current JADE message transport system has to be improved for mobile devices. Connection to mobile devices can be lost in any time so message buffering is needed. As illustrated in Figure 6, devices are often in a private network (e.g., in most cases when using a GPRS connection) and also many devices in a fixed network can be behind a firewall. For these cases, there should be a gateway for agent platforms. The messaging gateway is an optional component of the CASCOM communication architecture. It is only needed in

cases where mobile device has no public IP-address, but it can be used in other cases as well.

The CASCOM Messaging Gateway is a buffer for messages going to the agent platforms in mobile devices. The gateway does not address translations as the agent platforms using the gateway are expected to use the address of the gateway. That is, the agents situated in an agent platform in a private network will never use their private IP-addresses as their transport addresses, but instead use the gateway's address. Further, the messaging gateway is totally transparent to agents. For the time being, the address of the gateway has to be given as parameter to the CASCOM agent platform situated in a mobile device[‡]. This way it can be made sure that using the gateway is fully invisible to the agents, and the gateway does not have to parse ACL messages. The gateway forwards the messages (both directions) based on information found on message envelopes.

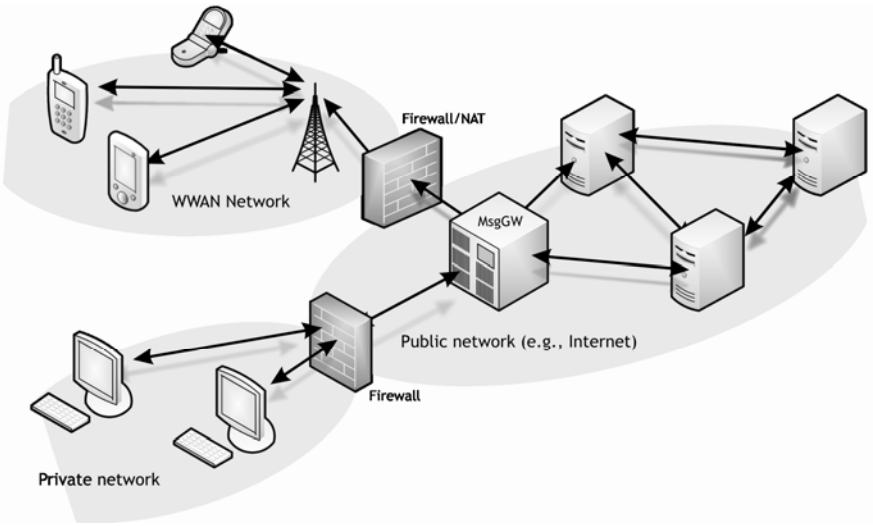

Figure 6: The CASCOM Messaging Gateway

Once the agent platform has established connection to the CASCOM Messaging Gateway, the (HTTP) connection must be left open so that the gateway is able to send messages to the agent platform which is behind a firewall. When the connection is closed (by the platform or because of unreliable wireless connection), the gateway leaves messages to the buffer waiting for the next time the connection is established. When the connection is established again by the same agent platform, the gateway must know whether it should deliver all the buffered messages or not. In the case the agent platform has been restarted and agents have not saved their state, there is no need to send the buffered messages and they can be discarded. The information whether buffered

[‡] This could be implemented also using some service discovery protocol. However, since the Messaging Gateway is an optional component of the CASCOM architecture, the implementation is kept as simple as possible.

messages should be delivered after reconnection is given by the mobile agent platform when opening the connection.

The messaging gateway does not buffer mobile-originated messages. Platforms use their own buffers for outgoing messages, but they send them through the gateway, which routes them to the destination. However, the protocol used between the CASCOM agent platform and the messaging gateway makes sure that no message is lost or duplicated in the case of unexpected wireless link disconnection. Details of this protocol can be found in [CA06b].

5 Service Discovery

The main service discovery in the CASCOM infrastructure happens in higher layers of the architecture [CA06a]. However, some support for it is also needed in the networking layer. Actually, the service discovery in the networking layer considers mainly the "low-level" service lookup in IP2P environments. This is realized by a directory system, WSDir. Its main functionality is to let heterogeneous service descriptions be registered and searched by certain clients. As such, it realizes a basic lookup function with basic retrieval schemes. To this end, coarse-grained text or keywords are used. This allows a pre-selection of services that subsequently will be used by agents in the service-coordination layer.

There are several main requirements for the directory system. First, it should be easy to invoke by any client. This led us to define a *Web Service interface* to the WSDir: it is a universally accepted standard, it provides a well-defined method to use the directory, and it allows for interacting with a heterogeneous set of clients. The sole requirement on the part of the client is that it should be able to communicate over a web service interface. Second, the nature of the applications to be realized requires the directory system to be distributed. We mainly consider a geographical specialization of the directories, as for the emergency assistance use case scenario (cf. Section 2). Finally, the construction of the network should induce minimal overhead and should be scalable; also, the network should be robust to changes in topology and the number of interactions with the system.

5.1 Service Discovery Agent

As the WSDir provides a Web Service interface and the CASCOM agents are using ACL for communication, we decided to develop a service discovery agent (SDA)[§]. The SDA receives service discovery requests from other agents and translates these to Web Service invocations to the WSDir. The SDA, however, is not only an intermediate between other agents and WSDir. Especially, the SDA is coupled with a semantic matchmaker agent (SMA) at the higher level of service coordination. The SMA can refine the selection of semantic web services. The Service Discovery and Service

[§] Other agents can found the SDA using FIPA service directories (i.e., directory facilitators)

Matchmaking functionalities are considered separately for reasons of efficiency and flexibility (for example, in some application domains the matchmaking functionality might not be necessary). The higher level service coordination is outside the scope of this paper. Details can be found in [CA06a]. Figure 7 depicts our service discovery architecture.

After the SDA has found a group of compatible services with a certain request, the SDA has two possible courses of actions: the first one is to return the result to the requester. The second one is that the SDA sends the discovered set of services to the SMA, as well as the original request. These steps are alternatives that are triggered by the use (or not) of the SMA. This means that the use of the matchmaker is not always required. For example, when the request has no inputs/outputs and/or pre-conditions and effects, the use of the matchmaker is not required, since its only task is to deal with these elements. If the matchmaking is needed, the SMA tries to do the matching (with the non elementary characteristics of the service) between the previously found descriptions (sent by the SDA) with the requester's specification. The obtained result is a set of service descriptions that satisfy the requester's needs. Finally, the SDA returns the results to the original requester. The service registration is similar to that of the service lookup, and thus not described here.

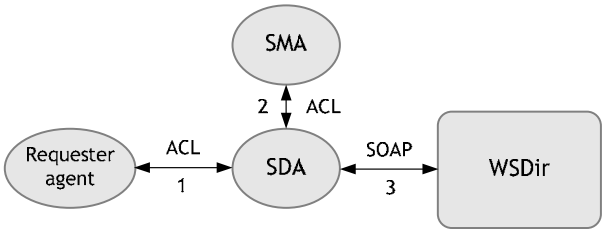

Figure 7: The service discovery process

5.2 Service Directories

A directory comprises a set of service entries which are managed by a collection of one or more directory services. All service entries are registered at a directory service as belonging to a specific directory. Such, directory services can form an arbitrary organizational structure (peer-to-peer, hierarchy etc.).

As noted above, directory services provide a Web service interface to a repository that holds service entries. The service entries in this store are all registered as belonging to a certain directory. The directory service forms the atomic unit of the directory federation. It allows clients to register, deregister, modify and search registrations in its repository. These registrations include service descriptions of services offered by clients as well as profiles of other directory service. By registering directory services in other directory service stores, the system becomes federated.

Figure 8 visually summarizes the relationship between service entries, directories and directory services. In the illustration, the directory service holds regular service entries

and a directory service entry belonging to a *Hospitals* directory as well as entries belonging to an *Insurers* directory. Both directories are contained in the *Body* directory, which in turn is contained in the all-encompassing "." directory.

Directory services employ *directory policies* to regulate the operation of directories. Policies are defined per directory service in the *directory service profile* and determine the behavior of a specific directory. Two types of policies can be distinguished: First, *Pro-active policies* are typically used for internal management of the directories. A policy may be attached to a directory to establish the number of times per hour data is propagated within the directory, how often old entries are removed and so on. Second, *reactive policies* assign a behavior to combinations of directories and various operations (register, search, modify, etc.). The policies are pre-defined and are executed whenever a bound operation is called.

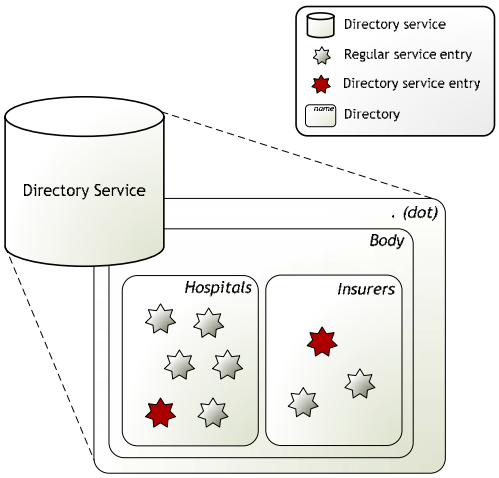

Figure 8: Visual recapitulation of directory system concepts

From the consequent application of policies, the network topology emerges. Policies can for example define how many entries can be registered per directory, which directories can be searched by which clients, and which types of services will be accepted. Each directory service can define its own policies or use one of the pre-defined policies. The only requirement on the part of a policy is that it can be executed. A straightforward example of a pre-defined policy is the *child/sibling* policy: this policy forwards all operations to both the known children (directory services registered in the service store) as well as its known siblings. The list of siblings is obtained by querying the parent directory service at which it has registered itself. As a directory service always has to register itself at another directory service, the policy is indifferent of the overall network topology.

6 Conclusions

In this paper, an innovative generic networking architecture has been proposed supporting intelligent peer-to-peer mobile service environments. This architecture includes agent execution environment for resource-poor mobile devices, support for agent communication over wireless communication path, and service discovery architecture. Service discovery is ensured through a federated directory system. All these together build a solid basis for developing business applications for mobile/wireless environments. Our main interest is to build an architecture that supports pervasive health care applications. Until now, we have implemented a prototype based on the presented networking architecture, focusing on the emergency assistance scenario described. At the end of the project, we will present with a fully fledged demonstrator for a concrete real-world business application scenario.

References

[BCP03] Bellifemine, F.; Caire, G.; Poggi, A.; Rimassa, G.; Jade — A White Paper. *EXP –in search of Innovation (TiLab Technical Magazine)*. 3(3):20–31. September 2003.

[CAS05] CASCOM Consortium. CASCOM Project Deliverable D3.1: Use Case Scenarios. 2005

[CA06a] CASCOM Consortium. CASCOM Project Deliverable D5.1: Distributed Service Directories for IP2P Environments. 2006.

[CA06b] CASCOM Consortium. CASCOM Project Deliverable D4.1: IP2P Network Architecture. 2006.

[FIP00a] Foundation for Intelligent Physical Agents. *FIPA ACL Message Representation in Bit-Efficient Specification*. Geneva, Switzerland. October 2000. Specification number SC00069.

[FIP00b] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Protocol for HTTP Specification*. Geneva, Switzerland, Oct. 2000. Specification number XC00084.

[FIP00c] Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification*. Geneva, Switzerland, Nov. 2000. Specification number XC00088.

[Hel03] Helin, H.: *Supporting Nomadic Agent-based Applications in the FIPA Agent Architecture*. PhD. Thesis, University of Helsinki, Department of Computer Science, Series of Publications A, No. A-2003-2. Helsinki, Finland, January 2003.

[HL02] Helin, H; Laukkanen, M.: Performance Analysis of Software Agent Communication in Slow Wireless Networks. In (Luijten, R.; Wong, E.; Makki, K.; Park, E.K): *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN'02)*, pages 354–361, Miami, Florida, USA. October 2002.

[LTL02] Laukkanen, M.; Tarkoma, S.; Leinonen, J.: FIPA-OS agent platform for small-footprint devices. In (Meyer, J-J.; Tambe, M.): Intelligent Agents VII, *Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), volume 2333 of Lecture Notes in Artificial Intelligence,* pages 447-460. Springer-Verlag: Heidelberg, Germany, 2002.

[MD00] Montenegro, G.; Dawkins, S.; Kojo, M.; Magret, V.; Vaidya, N.: Long thin networks. Request for Comments 2757, Jan. 2000.

[SRP04] Šišlák, D.; Rollo, M.; Pěchouček, M.: A-globe: Agent platform with inaccessibility and mobility support. In (Klusch, M.; Ossowski, S.; Kashyap, V.; Unland, R.): Cooperative Information Agents VIII, pages 199–214, 2004.