Proceedings of Second International Conference on Genetic and Evolutionary Computing, WGEC 2008, 25-26 September 2008, Jingzhou, Hubei, China, which should be cited to refer to this work.

BAF: A Bio-inspired Agent Framework for distributed pervasive applications

O. Brousse, G. Sassatelli, T. Gil, Y. Guillemenet, M. Robert, F. Grize[1]
E. Sanchez[2], Y. Thoma[2], A. Upegui[2], J.M. Moreno[3], J. Madrenas[3]
LIRMM, Univ. Montpellier 2/CNRS; 161 rue Ada, 34392 Montpellier, France.
e-mail: lastname@lirmm.fr

[1]Institute of Information Systems, Université de Lausanne (UNIL), Switzerland
[2]ReDS, HEIG-VD, Yverdon-les-Bains, Switzerland
[3]Department of Electronic Engineering, Technical University of Catalunya (UPC), Barcelona, Spain

***Abstract*—In this paper we introduce the Bio-inspired Agent Framework (BAF) developed within the Perplexus IST European project[1]. This BAF is FIPA (Foundation for Intelligent and Physical Agents) compliant as based on the JADE Multi-Agent Platform, portable and suitable for Ad-hoc networks of mobile nodes (MANET). Its bio-inspired capabilities and reliability services provide a powerful bio-inspired distributed tool that opens interesting perspectives for adaptive sensor networks.**

***Index Terms*—Bio-inspired, Scalability, Distributed platform, Ad-hoc Network, Multi-Agent Framework.**

I. Introduction

The Perplexus Project aims at developing a scalable and ubiquitous platform endowed with bio-inspired features to simulate complex phenomena.

The platform is composed of a cluster of communicating nodes called UBIDULES (UBIquitous moDULES). These nodes are built around a microprocessor and a specific reconfigurable integrated circuit that both provide support to distributed bio-inspired mechanisms. Furthermore, application-specific hardware sensors and actuators may be appended to any Ubidule for environmental interaction purposes.

A. Previous work on bio-inspiration

There exist several theories relating to life, its origin and all its associated characteristics. It is however usually considered that life relies on three essentials mechanisms that are Phylogenesis, Ontogenesis and Epigenesis (P, O and E in short):

1) *Phylogenesis* deals with the evolution of a set of species. Evolution gears species towards a better adaptation of individuals to their environment; genetic algorithms are inspired from this very principle of life.
2) *Ontogenesis* describes the origin and the development of an organism from the fertilized egg to its mature form. Biological processes like healing and fault tolerance are qualified of ontogenetic.
3) *Epigenesis* refers to features that are not related to the underlying DNA sequence of an organism. Learning as of performed by Artificial Neural Networks (ANN) is a process which scope remains limited to an individual lifetime and therefore is Epigenetic.

These three fundamental mechanisms have inspired researchers and led to the realization of bio-inspired artificial systems (be they software, hardware or mixed) that use up to two mechanisms. Some ANN implementations make use of Epigenesis (learning) and have been realized in several forms: dedicated digital circuits (such as proposed in [1] and [2]) or analog circuits [3]. Some other works combine 2 mechanisms like morphogenetic which is a PO combination proposed in [4]. In [5], an evolving ANN is introduced which can be considered as a PE combination (learning and evolution). Finally, in [6] variable topology ANNs are presented and can be considered as OE combinations (growth and learning).

The POEtic chip [7] [8] (realized within the confines of the POEtic European project [IST-2000-28027]), was the first chip implementing all three axis of life together, making it the most “living chip” ever created. The main objective of this project was to design a specific reconfigurable circuit well suited for the implementation of POE systems, taking advantage of their “natural”, intrinsic parallelism.

This chip, as depicted in figure 1, contains an embedded processor which runs genetic algorithms (Phylogenesis) and computes the fitness of individuals installed on an array of configurable processing elements named “molecules”.

These molecules are similar to FPGA slices and therefore rely on Look-Up Tables (LUTs), registers and reconfigurable datapaths. Molecules are grouped together into “cells” that carry out certain functionalities and may undergo processes such as migration, replication, etc.

[1][IST-2006-034632] web site : http://www.perplexus.org

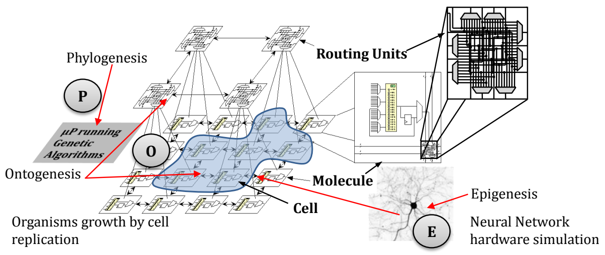
Phylogenesis
Routing Units
P
µP running Genetic Algorithms
O
Ontogenesis
Molecule
Cell
Epigenesis
Organisms growth by cell replication
E
Neural Network hardware simulation

Figure 1: POEtic chip overview.

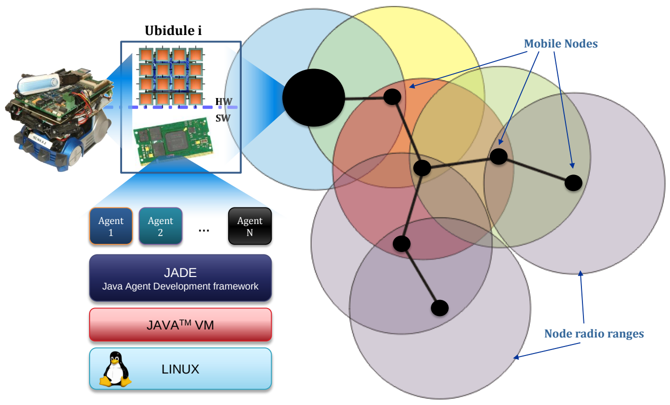

Figure 2: Perplexus platform overview.

B. *The Perplexus platform*

Based on this work, the goal of the Perplexus Project is to develop a scalable and distributed hardware platform endowed with bio-inspired capabilities. This will enable the simulation of large-scale complex systems and the study of emergent complex behaviors; this in a virtually unbounded wireless network of computing modules.

The Perplexus Ubidules integrate at least one bio-inspired chip (which has an architecture comparable to the POEtic chip) and an ARM microprocessor that is in charge of software aspects. Figure 2 gives an overview of the obtained platform which is represented in form of network of Ubidules. Each Ubidule is in turn represented as an unit made of a hardware and a software partition. An embedded Linux operating system runs on top of the microprocessor and handles most high-level functionalities.

Three applications are proposed within the project to demonstrate the advantages of a such platform. In this paper we focus on an embodied application (i.e. robotic application) that takes advantage of learning and evolution capabilities of the platform.

Among the many challenges that form this project, this paper puts focus on the work realized at the platform level for:

- Enabling the implementation of distributed applications that take advantage of the bio-inspired features of the platform. This work relies on an agent-based programming methodology (as suggested in Figure 2) and a programming framework made of resident agents which provide support for the three bio-inspired mechanisms presented previously.
- Ensuring communication reliability and scalability among Ubidules. The chosen communication medium being wireless and Ubidules being mobiles, appropriate software solutions have to be used for guaranteeing these two properties.

This paper is organized as follows:

- Section II introduces the Perplexus Platform Network level challenge and the proposed solution.
- Section III presents the FIPA compliant programming framework we use as a base for our BAF.
- Section IV presents the proposed solutions to create a POE dedicated BAF using previously presented solutions.
- Section V finally draws some conclusions on the realized work.

II. Network support

The modular structure of the Perplexus platform offers scalability due to the decentralized network structure which avoid central bottlenecks. Nevertheless it represents a challenge at the network level. This challenge becomes critical when nodes equipped with actuators (for example motors) are mobile.

In the literature such paradigm is known as MANET for Mobile Ad-Hoc NETwork. This IETF[2] working group[3] is in charge of proposing routing solution and standardizing IP routing protocols in the scope of wireless routing with either static or dynamic topologies.

MANET routing algorithms can be classified into two families:

- **Reactive MANET Protocols (RMP)** that search for a route between A and B nodes when a communication is intended between these nodes. AODV (for Ad-hoc On-demand Distance Vector) [9] and DSR (for Dynamic Source Routing) [10] are reactive protocols.
- **Proactive MANET Protocols (PMP)** which nodes regularly exchange messages in order to maintain all available routes up to date. OLSR (for Optimized Link State Routing) [11] is a proactive protocol.

A. *OLSR Protocol*

The OLSR routing protocol is among the most popular and effective solutions [12]. This proactive routing protocol regularly sends 3 different types of messages to create and maintain automatically network routes. This mechanism is well suited for most applications because it provides reduced communication latency, due to mostly

[2]IETF: Internet Engineering Task Force
[3]IETF MANET: www.ietf.org/html.charters/manet-charter.html

up to date routes (in comparison to reactive protocols that are slower to establish a route).

OLSR has been ported and tuned to the XScale platform. The chosen OLSR implementation offers the possibility to use plugins to provide additional services such as name/address translation.

For validating this solution we conducted several experiments which confirmed that proactive protocols such as OLSR perform better with respect to latency. Figure 3 shows the experimental protocol we used for OLSR. This map of the premises shows four nodes, three being static and the last one (Ubidule 3) in motion along the track (illustrated by the plain arrow on figure 3).

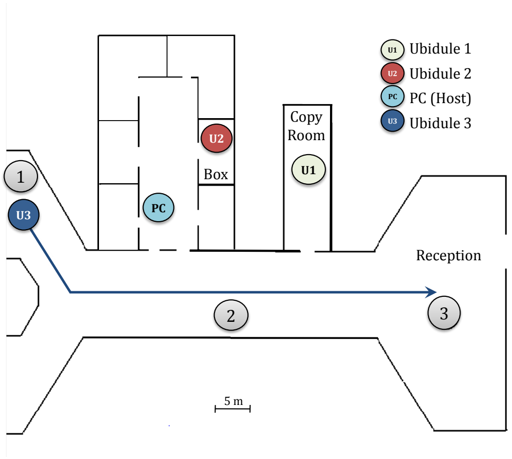

Figure 3: Mobile test protocol.

As suggested in figures 3 and 4, different network topologies are observed. Changes from one to another occur whenever a node drops out or cames in the radio range of another.

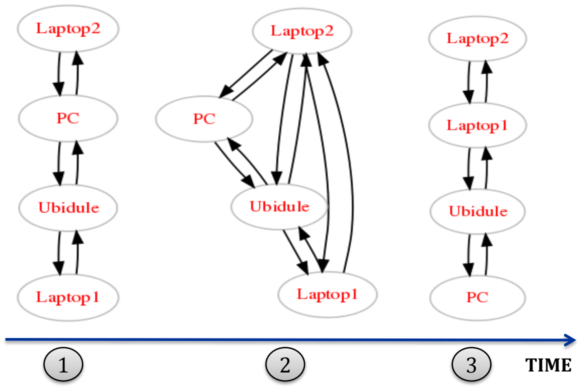

Figure 4: Time changing Topology.

Figure 5 shows the evolution of the bitrates received by the mobile node from the three other units; it can be clearly seen that a change in the network topology results in a break in the communication that lasts up to 5 seconds.

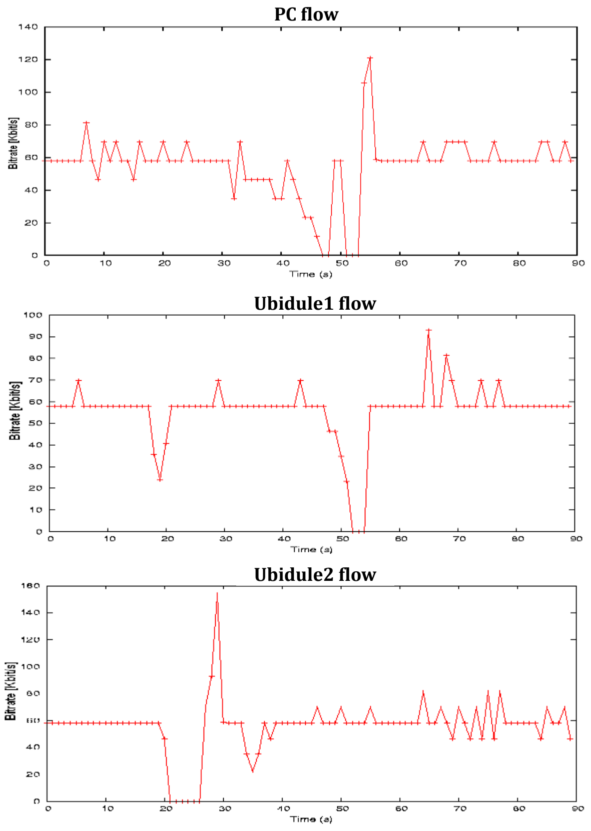

Figure 5: Mobile test: mobile node received message flows.

We consider these results satisfactory for the targeted applications; furthermore the flexibility of the chosen OLSR implementation allowed us to use a nameservice (DNS-like) plug-in that proves mandatory in the following. The nameservice plug-in acts in two successive steps:

1) The plug-in uses OLSR to broadcast messages containing IP and hostname information.
2) It collects other nameservice messages and stores received data in the hostsIP file (i.e. /etc/hosts for linux OS).

Consequently the nameservice allows each module to build a local routing table working with IP addresses and hostnames.

OLSRD and a slightly modified nameservice plug-in reveals to be a stable and efficient solution. In the case of our MANET application, this solution perfectly fits our needs and does not overload CPU with unnecessary functionalities.

III. The FIPA Multi-Agent System

The distributiveness and capabilities of the Perplexus platform greatly rely on the chosen programming style. Object-oriented programming (OOP) has become popular during the past decades mainly thanks to object-oriented languages like C++ which provided many benefits over other less formally defined languages.

Agent-oriented programming (AOP) derives from the initial theory of agent orientation which was first proposed by Yoav Shoham [13]. This programming format infers OOP by endowing objects with additional characteristics; they are viewed as entities which exhibit behaviours, capabilities and are entitled to take decisions. Table I summarizes the fundamental differences between AOP and OOP.

	OOP	AOP
Basic Unit	object	agent
Parameter defining state of Basic Unit	unconstrained	Beliefs, Commitments, Choices...
Process of computation	Message passing and response methods	Message passing and response methods
Type of messages	unconstrained	Inform, Request, Offer, Accept, Refuse
Constraints on methods	none	Honesty, Consistency...

Table I: OOP/AOP comparison

Agent-orientation was initially defined for promoting a social view of computing and finds natural applications in areas such as Artificial Intelligence or social behaviours modeling. An AOP computation consists in making Agents interact with each other through typed messages (i.e. constraint messages) of different natures: they may be informing, requesting, offering, accepting, and rejecting requests, services or any other type of information. Messages type is generally called a performative. AOP furthermore fixes constraints on the parameters defining the state of the Agent (beliefs, commitments and choices).

These constraints actually define the Agent Oriented computational system which is then viewed as a set of communicating software modules that exhibit a certain degree of awareness. These characteristics naturally geared the PERPLEXUS modeling framework towards AOP which fits perfectly the objectives of the platform.

A. FIPA: Foundation for Intelligent and physical Agent

The FIPA[4] defines standards for interacting Multi-Agent multi-Platforms systems. Since 2005, this organism joined the IEEE standardization.

Figure 6 shows the FIPA standard structure of an Agent Platform (AP). Three main services ensure FIPA platforms reliability and functionality:

- **AMS** (**A**gent **M**anagment **S**ystem) that is in charge of the platform's agent's lifecycle; it can create, suspend, resume or kill agents. Due to its main function AMS also provides a white page service containing all agents "living" on the platform.
- **DF** (**D**irectory **F**acilitator) that is in charge to provide a yellow pages service. This service associates an agent to its offered services and a service to agents that provide it.
- **MTS** (**M**essage **T**ransport **S**ystem) that provides all low-level communication functionalities. Therefore Agents can communicate with each other regardless of their location (same or different APs).

[4]http://www.fipa.org/

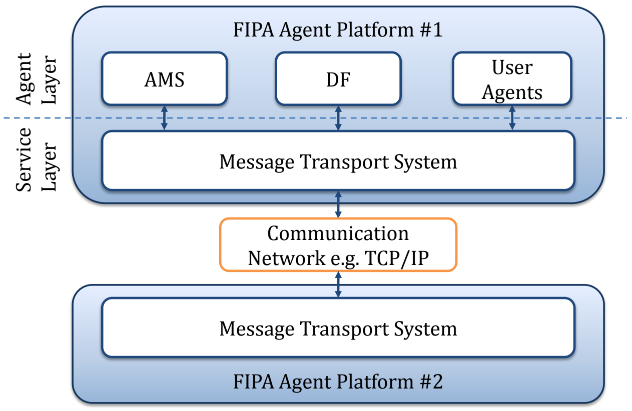

Figure 6: FIPA standard overview.

Figure 6 shows that AMS and DF are at agent level signifying that they are specific agents. On the contrary MTS is at a lower level meaning that it is a set of functions used by the agents. User agents sit at the same level as AMS and DF agents.

An important point to notice is that for now the FIPA standard does not include an AP search service. This drawback of the standard does not ease the use of multiple platforms as in our case. In this paper we propose a solution to address that point presented all along the following sections.

B. JADE: Java Agent DEvelopment Framework

There exists various Multi-Agent Platforms that allow to develop agent based applications, such as JADE [14], JXTA, FIPA OS, JAX or MADKIT [15].

JADE is java coded and therefore portable, FIPA compliant and exists in a lightweight version called LEAP[5] [16] that suits our hardware restrictions (PXA270) as it is running with Sun Microsystems J2ME[6] JAVA Virtual Machine.

A SUN phoneME advance JAVA virtual machine was ported to the XScale processor, for providing the necessary support for the Light Extensible Agent Platform (LEAP), the light version of JADE. LEAP has almost the same functionality as JADE but is designed for handheld devices making it appropriate for our platform.

Agents in a JADE Framework "live" in containers. These containers exist inside or outside of the original hardware hosting the JADE platform. JADE provides support for both communication scenarios:

[5]Light Extensible Agent Platform
[6]J2ME: Java2 Micro Edition

1) **intra-platform**, where agents are registered on the same platform and are considered to be local even if they are hosted on remote hardware (left part of figure 7).
2) **inter-platform**, where agents are registered on their platform, services discovery allows Agents to find other agents to work with (right part of figure 7).

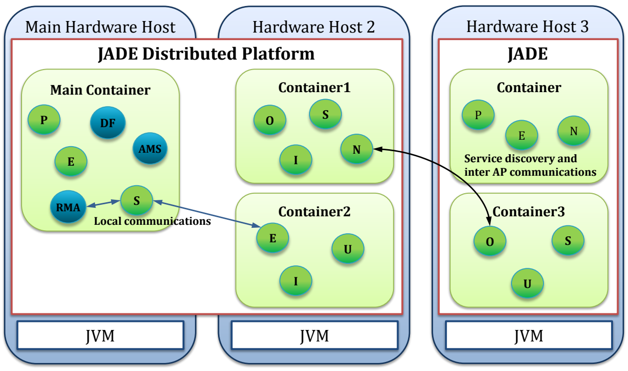

Figure 7: JADE programming platform.

In both cases, unless agents are hosted on the same AP and the same hardware, inter hardware messages are transmitted through the TCP protocol.

JADE APs communicate with each other over TCP/IP using Message Transport Protocols (MTP). In our case we decided to use HTTP MTP in order to ease AP communications and unify AP name and address with hardware hostname. This point is discussed in the following section.

As shown in figure 7, an optional specific agent called RMA (Remote Management Agent) allows user to visualize the current state of the JADE local AP and connect to remote APs.

IV. BAF Biomimetic capabilities

Previously described solutions are used as foundations to develop a Bio-inspired Agent Framework (BAF) suitable for distributed and decentralized platforms. This section focuses on two fundamental aspects of this BAF:

1) Description of the BAF and overview of the provided functionalities.
2) Description of the POE specific Agents.

A. *BAF overview*

Figure 8 shows the additional features of the BAF (grey-shaded areas).

This Framework adds a low level service layer that comprises the Ad-hoc networking features of the BAF, depicted in figure 8. This layer includes OLSRD and the nameservice plug-in. The hostname/IP table (periodically updated by the nameservice) can easily be accessed by other software entities such as JADE agents. These functionalities are accessed by any agent through a specific agent called the *Network agent*.

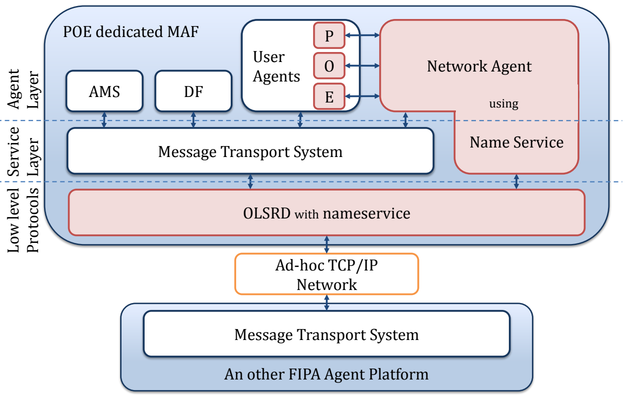

Figure 8: POE dedicated BAF overview.

This agent is a mandatory agent in a BAF platform. It enables platform modules interactions and ensures the overall platform reliability. As this particular agent provides AP level services and low-level functionalities (such as message broadcasting) it spans on both highest level layers of the diagram.

The use of the HTTP Message Transport Protocol allows deducing the AP name and address making a JADE AP capable of establishing communications with one another through the ad-hoc network. Figure 9 describes this peer discovery mechanism.

Once the nameservice has edited the system Hostname/IP file (step 1), the Network Agent is able to create the peer platform list (step 2). Other Agent lists can be created in the same way; all mandatory platform agents are listed in the network agent and can be joined transparently.

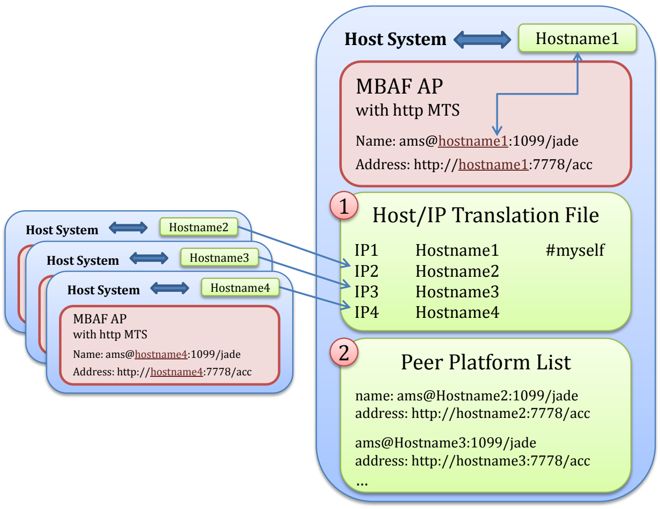

Figure 9: BAF AP addresse deduction

An Host agent has been designed to provide a single interface for the platform controlling. This Agent is able to remotely schedule applications from a host station thanks to the Network agent services.

Using the peer discovery service capabilities, a local application agent can initiate a service search not only on its own platform -using the local DF- but on all reachable DF agents thanks to the Network agent. This “ServiceSearch” service eases the development of fully distributed application on the BAF.

At a lower level, the ServiceSearch service uses another functionality of the Network Agent that also provides a broadcast message service. Such service allows application agents or host agents to send messages to a set of agents without knowing their names or addresses. Figure 10 shows the three main steps of this protocol taking the example of a Host agent broadcasting orders to the platform.

The first step is a local request of the Host to its local Network Agent for broadcasting a given message to a given set of Agent here Epigenetic Agents (described later on). Then the first Network Agent broadcasts this message to others Network Agents (second step), the message is then transmitted locally to the target agents.

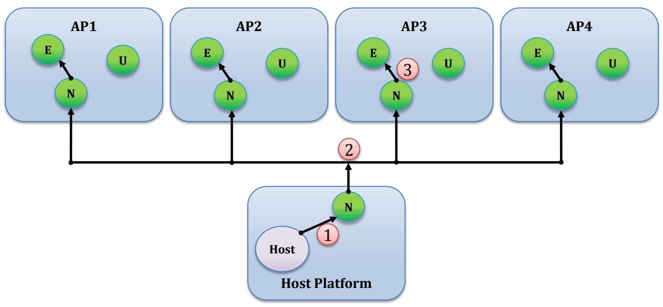

Figure 10: Broadcast message service.

This protocol is mainly used for sending global orders to the platform such as global “servicesearch” “Start Application” “Stop Application” or “Switch Mode” (passing from software mode to hardware mode of the Perplexus platform). In this case Network agents are final receivers of order messages. The main advantage of this method is that the Host Agent and the user it represents does not need to know addresses of all final receivers.

B. POE dedicated features

Three specific agents are represented in the “User Agents” box in Figure 8. These P, O and E agents implement respectively Phylogenetic, Ontogenetic and Epigenetic processes.

Table II reports the possible hardware implementation of these agents in the specific frame of the Perplexus Platform. As hardware implementation requires the Perplexus bio-chip, hardware that differs from Ubidule are not able to support it. These hardware features are designed to speed up simulation of POE applications and more generally complex systems. The following sections describe these three Bio-inspired agents.

	SW	HW
Phylogenetic agent(s)	√	
Ontogenetic agent(s)	√	√
Epigenetic agent(s)	√	√

Table II: Agents operating mode

1) P agent: The P agent has no possible hardware implementation because it is in essence a distributed genetic algorithm. This agent is responsible of evaluating the fitness of the individual embodied on the Ubidule. Selection rules, crossover and mutation processes are application-specific and eventually lead to the installation of a new generation on the Ubidule population.

2) O agent: The software implementation of the Ontogenetic agent is actually in charge of creating other software agents with given parameters. This represents the growth capabilities of the system.

The hardware implementation of this agent operates with the same objectives but instead of creating software agents, it uses runtime partial reconfiguration capabilities of the bio-chip to install functionalities on the available hardware functional units.

In both cases (software agents and hardware functional units) the O agent may create an Epigenetic Agent.

3) E agent: In the Perplexus project Epigenetic processes are viewed as Artificial Neural Networks. Therefore the E agent implements ANN either in software or in hardware. Some Neural Networks are available ranging from MLP with backpropagation to Spiking Neural networks, other ANN types can be added if needed.

C. Validation application

In order to prove the reliability of the platform, a simple validation application based on a race of robotic toys was developed. The used robot is depicted in Figure 2. Due to the early development of the POE dedicated hardware, this application uses full software simulation mode as depicted on figure 11.

P, O and E software agents provide POE support for the control of the robots. P agent is responsible of the robot behaviour evolution (generation management and application high level management). Ontogenetic agent instantiates an ANN the Epigenetpic agent) based on the genome provided by the P agent. This Epigenetic agent is a simple feed-forward ANN which reads binary information from three proximity sensors installed on the front, front-left and front-right sides of the robots, and issues speed orders to the two motors. Figure 11 shows, next to the P, O, E and N agents two additional agents: the Interface agent (I agent) that acts as a wrapper for any agent of the platform to communicate with the sensors and actuators; and the Ubicom agent (U agent) that handles communications with the Ubichip (unused here).

For this application robots are moving into a closed arena containing obstacles and a start/finish line. Robots

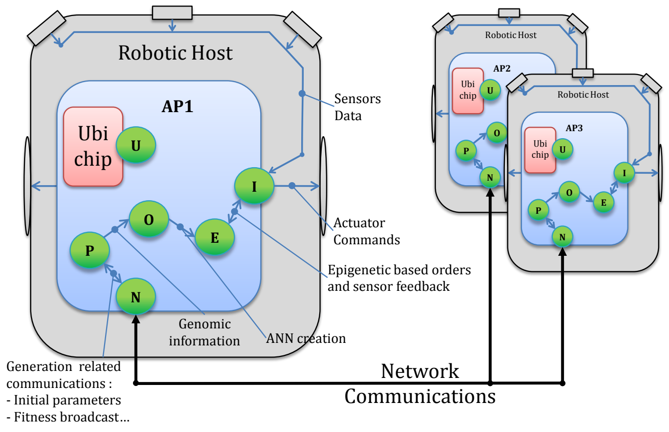

Figure 11: Software application example.

are asked to run one lap of this track. Figure 12 shows the principle of this genetic race -the laptime gives the fitness of a given robot. These agents are crossed and/or mutated to create the next generation replacing inadequate behaviours. Generation after generation robots exhibit smarter behaviours proving the reliability of the software and the possibility to handle POE problems via the platform. A demonstration video is available online at: http://www.lirmm.fr/~brousse/Ubibots.

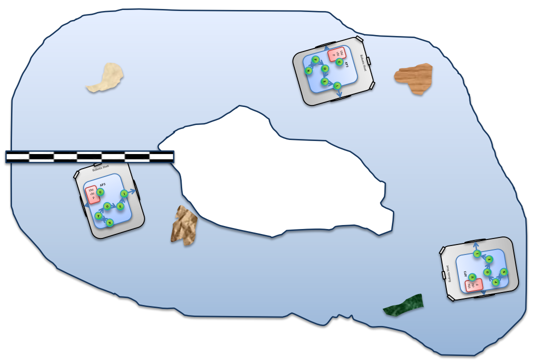

Figure 12: Arena overview.

V. Conclusion

This paper presents some of the work realized within the confines of the Perplexus Project. These contributions provide a reliable middleware to support both bio-inspiration and agent-oriented programming for distributed pervasive platforms. The proposed BAF has been designed with broader application fields in mind and should prove appropriate for many sensor-network applications where adaptability brings advantages.

This software environment has been tested through a POE robotic application that proves the reliability of the platform with satisfying result concerning communication and basic POE computation. The future availability of the bio-inspired integrated circuit will soon help demonstrating the combined advantages of hardware support and pervasiveness for distributed platforms. Future work relies on enabling transparent migration between JADE software agents and their embodied hardware counterparts.

Acknowledgements

This project is funded by the Future and Emerging Technologies programme IST-STREP of the European Community, under grant IST-034632 (PERPLEXUS). The information provided is the sole responsibility of the authors and does not reflect the Communitys opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

[1] T. Shibata and T. Ohmi, “Neuron MOS binary-logic integrated circuits. i. design fundamentals and soft-hardware-logic circuit implementation,” *IEEE Transactions on Electron Devices*, vol. 40, no. 3, pp. 570–576, Mar. 1993.

[2] T. Shibata, T. Nakai, Y. N. Mei, Y. Yamashita, M. Konda, and T. Ohmi, “Advances in neuron-MOS applications,” in *IEEE International Conference on Solid-State Circuits. Digest of Technical Papers. 43rd ISSCC*, 1996, pp. 304–305.

[3] S. Eberhardt, T. Duong, and A. Thakoor, “Design of parallel hardware neural network systems from custom analog VLSI ‘building block’ chips,” in *International Joint Conference on Neural Networks (IJCNN)*, vol. 2, 1989, pp. 183–190.

[4] D. Roggen, D. Floreano, and C. Mattiussi, “A Morphogenetic Evolutionary System: Phylogenesis of the POEtic Tissue,” in *Proc. of the 5th Int. Conf. on Evolvable Systems (ICES 2003)*, A. M. Tyrrell, P. C. Haddow, and J. Torresen, Eds. Springer-Verlag, 2003, pp. 153–164.

[5] S.-W. Moon and S.-G. Kong, “Block-based neural networks,” *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 307–317, Mar. 2001.

[6] A. Perez-Uribe, “Structure-adaptable digital neural networks,” Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 1999.

[7] G. Tempesti, D. Roggen, E. Sanchez, Y. Thoma, R. Canham, A. Tyrrell, and J. M. Moreno, “A poetic architecture for bio-inspired hardware,” *Conference on the Simulation and Synthesis of Living Systems (Artificial Life VIII)*, pp. 111–115, dec 2002, sydney, Australia,**[IST-2000-28027]**. [Online]. Available: www.poetictissue.org

[8] Y. Thoma, “Tissu numérique à routage et configuration dynamique,” Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 2005.

[9] E. M. R. Charle E. Perkins, “Ad-hoc on-demande distance vector,” Dec. 1998.

[10] D. Johnson and D. Maltz, “The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4,” Feb. 2007.

[11] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, “Optimized link state routing protocoll for ad-hoc networks,” 2001, INRIA Roquencourt, HiPERCOM project.

[12] T. Clausen, P. Jacquet, and L. Viennot, “Comparative study of CBR and TCP performance of MANET routing protocols,” *Workshop MESA*, 2002, INRIA Roquencourt, HiPERCOM project.

[13] Y. Shoham, “Agent oriented programming,” *Journal of Artificial Intelligence*, vol. 60, 1996.

[14] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. Wiley, April 2007.

[15] G. Nguyen, T. Dang, T, L. Hluchy, M. Laclavik, Z. Balogh, and I. Budinska, “Agent platform evaluation and comparison,” 2002, slovak Academy of Sciences, Institute of informatics, Pellucid 5FP IST-2001-34519.

[16] J. Lawrence, “LEAP into Ad-Hoc Networks,” *ACM Workshop on Agents in Ubiquitous and Wearable Computing, AAMAS*, 2002.