Decentralized Lightweight Communication Platform for Heterogenous Multi-context Systems (Extended Abstract)

Michal Vince[1]* Michal Čertický[1]† Vladimír Dziuban[1]‡

Katedra aplikovanej informatiky, FMFI UK, Mlynská Dolina 842 48 Bratislava

In heterogeneous multi-context systems (MCS), a number of components, running on separate machines and implemented in different languages, need to cooperate and communicate with each other. Standards for such interoperability are defined in FIPA [FIPA, 2000-A] specifications. Majority of FIPA implementations, however, involve overly complex, mostly Java-based platforms, offering different services and hosting multiple agents. To facilitate simpler and more agile development of small agents in different languages, we present a lightweight decentralized communication platform.

Agents implemented in our framework are standalone processes, that communicate via FIPA ACL based messages [FIPA, 2000-B], transported through a RESTful peer to peer TCP/IP connections. They are identified by unique *agent name* and a list of *services* they provide.

Even despite the lack of any central server, the framework provides an implicit *agent management system* [FIPA, 2000-C] based on a discovery service that allows each agent to identify other agents on the local network along with the services they provide. Communication with agents outside of local network is achieved through the use of *gateway agents*.

From the implementation viewpoint, our framework consists merely of the set of libraries allowing any process to become a part of such MCS. These libraries are available for Python and C++ at the moment, while Java implementation is planned.

Following example implements MCS organizing the seminars of a research group. Individual agents run on different devices/programming languages, taking advantage of our platform's properties.

Example 1. *The system consists of:*

- *a set of **personal** Java-based agents (**PER**) running on the mobile phone/PDA of every group member, providing information about him,*
- ***sensoric** C++ agents (**SEN**) that use web cameras to observe certain rooms at the university,*
- ***timetable** agent (**TIM**) that provides access to the university timetable and room reservations,*
- *and a logic-based **scheduling** agent (**SCH**) written in Python as a frontend to a logic formalism such as ASP solver.*

*Now imagine, that one of the group members wants to organize a meeting. He orders his **PER** to arrange this meeting. **PER** then contacts all the other **PER**s, and finds out (using the GPS on their devices) which of them are currently out of town. If the sufficient number of them are available, he asks the **SCH** for the most appropriate time and place for the meeting. The **SCH** collects information from the **SEN**s to find out which rooms are empty and also checks if they aren't reserved for the next 2 hours. After receiving this information, the invitation can be sent to all the **PER**s of available members of the group.*

In future, we plan to make every agent capable of starting a gateway service whenever needed, thus maintaining the structural integrity, and improving the durability of MCS by automatic reorganization.

References

[FIPA, 2000-A] Foundation for Intelligent Physical Agents. FIPA Standard Specification. http://www.fipa.org/repository/standardspecs.html. 2000.

[FIPA, 2000-B] Foundation for Intelligent Physical Agents. FIPA ACL Message Structure Specification. http://www.fipa.org/specs/fipa00061/index.html. 2000.

[FIPA, 2000-C] Foundation for Intelligent Physical Agents. FIPA Agent Management Specification. http://www.fipa.org/specs/fipa00023/index.html. 2000.

*vince.michal@gmail.com

†certicky@fmph.uniba.sk

‡dziuban@ii.fmph.uniba.sk