The Thirteenth International Symposium on Artificial Life and Robotics 2008(AROB 13th '08),
B-Con Plaza, Beppu, Oita, Japan, January 31-February 2, 2008

# Dynamic Control of a Robotic Swarm using a Service-Oriented Architecture

Vlad M. Trifa[1,4*] Christopher M. Cianci[2] Dominique Guinard[3,4]
[1]Institute for Pervasive Computing, ETH Zurich, Switzerland
[2]Swarm-Intelligent Research Group, École Polytechnique Fédérale de Lausanne
[3]Auto-ID Labs ETH Zurich, 8092 Zurich
[4]SAP Research CEC Zurich, Kreuzplatz 20, 8008 Zurich

## Abstract

The development, deployment, and control of groups of robots is a tedious process even for experienced roboticists. Particularly in heterogeneous systems a high granularity of control and visibility is difficult to achieve. The lack of standardized interfaces and communication protocols to interconnect robots from different manufacturers makes it very difficult to develop flexible robotic applications. We propose an efficient system suited to support heterogeneous robotic swarms that can be used as a toolkit for fast prototyping of robust distributed applications. This system offers a flexible interface allowing external users to remotely control the swarm over the internet by using standardized communication protocols such as Web Services.

## 1 Introduction

Given the ubiquitous nature of the internet both in private and industrial settings, more and more processes are shifting from fully centralized to distributed solutions. To attain the required degree of flexibility and interoperability, the Service-Oriented Architecture (SOA) paradigm has been introduced and has become a common method for data exchange between large industrial enterprises. The key idea behind SOA is that different companies offer their services in the form of modular, and loosely coupled software components exchanging data over HTTP which can be consumed by other companies through the internet with no human intervention.

Telerobotics is no exception to this massive decentralization of industrial processes, and several projects where robots can be controlled over the internet have emerged in the last decade, as for example the Robotic Garden [4] and KhepOnTheWeb [9], among others.[1]

*Corresponding author: vlad.trifa@ieee.org

[1] http://ranier.hq.nasa.gov/telerobotics_page/realrobots.html

In the industrial sector, remote control of manufacturing processes can also be distributed, but in many cases a reliable communication channel is required to ensure functionality. Usually, industrial robotic teams (e.g. a manufacturing production line) are programmed assuming no equipment failure, and the control process is centralized on a single machine which orchestrates the synchronization between different tasks and robots. However, if a robot fails, it can take a lot of time (and money) to reconfigure a new robot and integrate it in the chain to resume the task. To avoid such losses, a powerful and robust infrastructure that offers full control over a group of heterogeneous robots through the internet can be very efficient solution.

Unfortunately, most telerobotic projects have been mainly focused on the control of a single robot. Our approach is relatively novel in that we are interested in providing a solid tool for experts working with *distributed* robotics. The metaphor of Swarm Intelligence [2], offers several promising methods for creating robust distributed systems. Global a priori knowledge of the system is not required, and coherent behavior emerges from local interactions between robots. Thus no centralized controller is needed, increasing the robustness of the whole system. However, the difficulty of properly analyzing such systems often prevents them from being implemented in industrial settings, hence the lack of appropriate standards for the associated tools and methods.

Specifically, when dealing with heterogeneous groups of robots, the principal challenge is to seamlessly interconnect devices made by different manufacturers and having different capabilities. Several different approaches have been proposed as potential standards; Web Services (WS), as a particular implementation of an SOA, have been selected here for their flexibility and its demonstrated success in other domains. Other SOA standards such as Universal Plug and Play (UPnP) have also been used to interconnect robots [1], but UPnP is not particularly well suited to

this task as it requires specific libraries that are not available on all operating systems.

To maximize compatibility, a fully open set of standards is preferable, as presented in WS, and demonstrated in various research efforts. A remote monitoring and control architecture based on the WS model [6], a control platform for a robotic arm [7], and a method to control several robots using WS in conjunction with Manufacturing Messages Specifications [10] have all been proposed. In addition, RoboLink [8] is an attempt supported by several industrial partners to standardize robust inter-robot communication protocols. Unfortunately, it is mainly concerned with the low-level communication specifics, and thus cannot easily be extended to other devices.

We advocate the use of WS for communication between the different components of the system (in particular the hardware and the user interface). A key advantage of using WS to access and control the robots is that they enable clients on diverse platforms (e.g. web pages, mobile phones, etc.) to have the same level of control over the network. In addition, WS provide a reliable and secure communication channel, automatic discovery of new devices, and an efficient publish-subscribe event notification mechanism.

In this project, we have built a central server based on Java Enterprise Edition (J2EE) that acts as a gateway between the end-user and the robot group. The server implements a WS that allows users to retrieve real-time information about the current status of the system as a whole, or of an individual robot. An advantage of this approach is that a control application can leverage both the centralized aspect of the server (which has full control over the robots) and the fully distributed nature of the robotic swarm.

## 2 Tools and Methods

The basic components of the system are shown in Fig. 1, and this section will described each of these parts more in detail.

### 2.1 Physical Device & Gateway Layers: The e-puck Robot

In this case study, the physical devices used were e-puck robots. The e-puck[2] is a miniature open-hardware robotic platform recently developed at the École Polytechnique Fédérale de Lausanne (shown in Fig. 2). The standard e-puck has eight infrared proximity and light sensors, a trinaural microphone array, a speaker, a three-axis accelerometer, and a Bluetooth interface for programming. It can also be fitted with custom pluggable modules that stack in between the two standard boards, such as the short-range radio communication turret used here [3], which provides a subset of the 802.15.4 and ZigBee protocols and is fully interoperable with the MicaZ [5] nodes used here as base stations in the physical gateway layer.

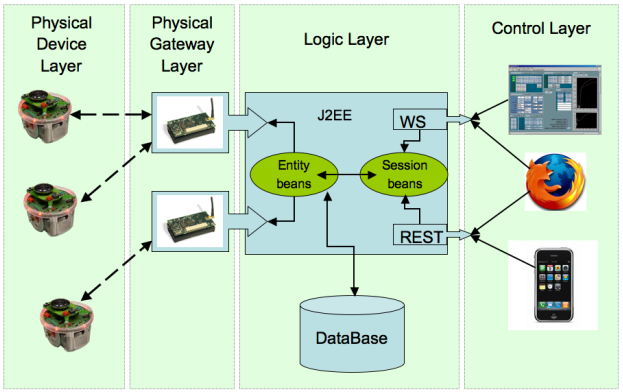


Figure 1: The general architecture of our system is composed of four parts. From left to right: The *Physical* layer, consisting of the actual robots; the *Gateway* layer, which is the connection between the physical devices and the system; the *Logic* layer, containing the server that logs the data coming from the devices; and the *Interface* layer, which includes any device or interface for an external user or users.

### 2.2 Logic Layer: Web Services

Web Services are a set of standards[3] similar to Remote Procedure Calls (RPCs), where the functionality of different software components is exposed as an API with different methods that can be invoked on the service. These functional specifications allow different software modules to collaborate and exchange data regardless of the underlying hardware and software platforms.

From a functional point of view, our system is composed of three disjoint components. As can be seen in Fig. 1, the end-user is only involved at the Interface Layer; the Human-Computer Interface to the robotic swarm. The Interface Layer communicates directly with the Logic Layer (the server), which contains the main application and is responsible for the transport of

[2] http://www.e-puck.org

[3] http://www.w3c.org/2002/ws/

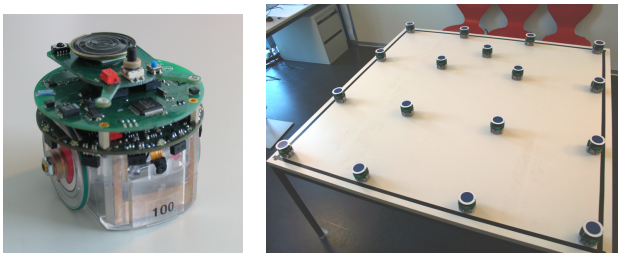

Figure 2: LEFT: The e-puck, a small-scale experimental robotic platform. Shown here with the radio communication board stacked between the basic module and the jumper board, allowing the implementation of sensor networks and other networked robotic systems. RIGHT: Several e-pucks in a table-top arena (Shown with colored markers attached to facilitate tracking by an overhead camera).

commands and data between the user and the physical devices. The Logic Layer is implemented using J2EE, which is a powerful, industrial development and production environment. The choice of J2EE was based on its reputation as a robust environment that supports industrial applications, and Web Services in particular.

### 2.3 Control Layer: User Interface

The development of distributed robotic applications often requires significant additional effort to design custom user interfaces for the visualization, configuration, and control of a robotic swarm. In addition, these interfaces can frequently be very application-specific. One feature of this system is that, using the WS standard, almost any physical device or piece of software capable of implementing HTTP (e.g. a stand-alone Java application, PDA, Tablet PC, mobile phone, etc.) can be used to interact with the multi-robot system, regardless of the operating system or programming language on the device. The service provider needs only to publish a list of the different functions that can be called by the client, allowing any user to create a personalized user interface.

For devices with limited computational power (e.g. a GPRS-enabled mobile phone), an additional Representational State Transfer (REST)[4] interface has been developed. REST allows clients to interact with the system simply by accessing specific URLs that encode function calls and their associated parameters.

[4] http://rest.blueoxen.net/

## 3 Experimental Case Study

As a proof of concept, we used a team of e-puck robots capable of performing various actions (e.g., illuminating LEDs, setting motor speeds, reading proximity sensors, azimuth estimation of incident sound, etc.). If combinations of these actions are considered 'tasks,' the following scenario can be used to illustrate one possible use case for the proposed architecture.

The e-pucks are able to perform three tasks ($T_1$, $T_2$, and $T_3$). Two e-pucks are executing $T_1$, and a third is executing $T_2$.

1. A fourth e-puck is added to the swarm, and sends an INIT message to the server.
2. The server acknowledges the robot by sending it a command to execute $T_2$.
3. The user changes the wheel speed of an e-puck using the web interface.
4. All LEDs of another robot are turned on using a Java application.
5. All e-pucks are ordered to execute $T_3$ using a mobile phone.

This sample sequence is a basic illustration of the different commands that can be send to the robot team. Of course, given the nature of our system, much more elaborate sequences can be executed from remote locations.

The server is responsible for keeping track of the status of all connected devices, and logging all received data into a database. Additionally, it also generates the actual commands that are sent to the robots though different physical gateways. When a robot is turned on, it broadcasts a message which specifies its physical address and available capabilities. The server will acknowledge the new arrival by sending a command specifying a behavior to execute based on the current state of the system, and task priorities.

## 4 Conclusion and Future Work

We have presented an infrastructure for easy development and control of distributed robotic applications. Our approach focuses on the runtime configuration (and reconfiguration) of the robots; different devices can perform their tasks autonomously, accept direction from a central entity, or some mixture of both modes. Another novel aspect of the system is its use of WS for communication between the end-user and the

system, allowing personalized control interfaces (created using any programming language on any hardware platform, provided that HTTP is supported).

This simple case study also illustrates how the WS standard can significantly increase the interoperability of heterogeneous sensor and actuator networks. In addition, integration of any of the available features in WS standard is relatively straightforward. For example, secure and reliable communication can be established with minimal network overhead, and new robots can be automatically discovered and integrated into the system. As the WS standard is becoming increasingly common in industrial enterprises, the merging of robotic and business applications—a growing demand—is greatly facilitated.

For the current case study, the WS protocol was not implemented directly on the robots, due to the limited computational resources on the e-pucks (this formality was handled in the Physical Gateway layer), but Moore's law may soon make this relatively feasible. However, it will nonetheless be essential to compare the performance of WS-based communication with existing proprietary protocols, and carefully find the balance between performance and flexibility for each application. A testbed composed of industrial robots will also be needed to quantitatively evaluate the performance of the architecture presented here.

Future work also includes efforts to adapt the automated service discovery mechanism to handle arbitrary device types. In particular, methods to encode different capabilities of robots are to be explored (e.g., an XML description of a robot's sensors, actuators, capabilities, etc.), and also automatically generated user interfaces based on the specific nuances of each robot.

Clearly, the work presented here only serves as an initial prototype for a robust and flexible infrastructure that can support the development of extremely large distributed robotic systems. A fully functional system (where performance issues can be systematically analyzed) will need careful design and optimization based on the tasks the robots will perform, and the trade-off between flexibility and reliability must be carefully evaluated. However, the fundamental principles proposed here represent a scalable solution which will be able to meet industrial standards.

## Acknowledgments

The authors would like to thank the European Commission and the partners of the European IST FP6 project Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices (SOCRADES; http://www.socrades.eu) for their support. Christopher Cianci is currently sponsored by a grant from the Swiss National Science Foundation (Contract Number PP002-116913).

## References

[1] Sang Chul Ahn, Jung-Woo Lee, Ki-Woong Lim, Heedong Ko, Yong-Moo Kwon, and Hyoung-Gon Kim. Upnp sdk for robot development. In *SICE-CASE International Joint Conference*, 2006.

[2] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence From Natural to Artificial Systems*. Oxford University Press, New York, NY, 1999.

[3] Christopher M. Cianci, Xavier Raemy, Jim Pugh, and Alcherio Martinoli. Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In *Simulation of Adaptive Behavior (SAB-2006), Swarm Robotics Workshop*, pages 103–115, Rome, Italy, October 2006, Lecture Notes in Computer Science (2007), vol. 4433.

[4] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley. Desktop tele-operation via the world wide web. In *In IEEE International Conference on Robotics and Automation (ICRA 1995).*, 1995.

[5] Jason L. Hill and David E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, November/December 2002.

[6] Min-Hsiung Hung, Kuan-Yii Chen, and Shih-Sung Lin. Development of a web-services-based remote monitoring and control architecture. In *In IEEE International Conference on Robotics and Automation (ICRA 2004).*, volume 2, pages 1444–1449, April/May 2004.

[7] Bong Keun Kim, M. Miyazaki, K. Ohba, S. Hirai, and K. Tanie. Web services based robot control platform for ubiquitous functions. In *In IEEE International Conference on Robotics and Automation (ICRA 2005).*, 2005.

[8] Masahiko Narita, Makiko Shimamura, and Makoto Oya. Reliable protocol for robot communication on web services. In *Proceedings of the International Conference on Cyberworlds (CW'05)*, 2005.

[9] P. Saucy and F. Mondada. Khepontheweb : One year of access to a mobile robot on the internet. In *Proc. of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS1998*, 1998.

[10] Bin Wu, Bing-Hai Zhou, and Li-Feng Xi. Remote multi-robot monitoring and control system based on mms and web services. *Industrial robot: an international journal*, 34(3):225–239, 2007.