

Programming Modular Robots with the TOTA Middleware

Marco Mamei

Dipartimento di Scienze e Metodi dell'Ingegneria
University of Modena and Reggio Emilia
Reggio Emilia, ITALY

marco.mamei@unimore.it

Franco Zambonelli

Dipartimento di Scienze e Metodi dell'Ingegneria
University of Modena and Reggio Emilia
Reggio Emilia, ITALY

franco.zambonelli@unimore.it

ABSTRACT

Modular robots represent a perfect application scenario for multiagent coordination. The autonomous modules composing the robot must coordinate their respective activities to enforce a specific global shape or a coherent motion gait. Here we show how the TOTA ("Tuples On The Air") middleware can be effectively exploited to support agents' coordination in this context. The key idea in TOTA is to rely on spatially distributed tuples, spread across the robot, to guide the agents' activities in moving and reshaping the robot. Three simulated examples are presented to support our claims.

Categories and Subject Descriptors

D.3.3 [Programming Languages]: Language Constructs and Features; C.2.4 [Distributed Systems]: Distributed Applications

General Terms

Modular Robots, TOTA Middleware

Keywords

Coordination, Modular Robots, TOTA Middleware

1. INTRODUCTION

A modular (or self-reconfigurable) robot is a flexible robot made up of a collection of (typically simple) autonomous elements connected with each other with some degree of freedom in their relative movements [2, 3].

The key idea underlying a modular robot is to have its components execute distributed control algorithms so as to coordinate their actions and let the robot assume a specific shape or move according to some specific motion pattern (i.e., gait).

Modular robots represent a perfect application scenario for multiagent systems. On the one hand, the modules con-

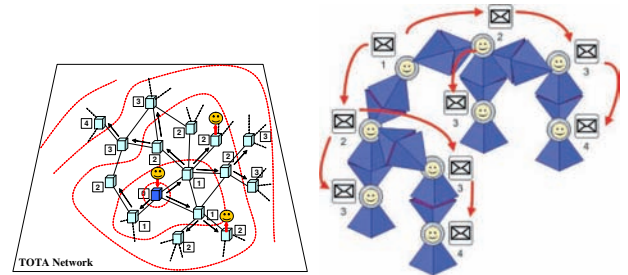


Figure 1: (left) The general scenario of TOTA. (right) TOTA in modular robots.

stituting the modular robot should be autonomous and running agent applications. This, in fact, avoids single point of failure and bottlenecks. If one module breaks down the other can reorganize their activities leaving the broken element behind. Autonomy enables also disconnected operations: modules can disassemble, move along different paths and reassemble subsequently. On the other hand, the main task the modules have to undertake is *coordination*. The agents governing the modules must coordinate their respective activities to enforce a specific global shape or a coherent motion gait in the whole robot (i.e., multiagent system).

For the purpose of supporting agent coordination, in our research, we developed a general middleware called "Tuples On The Air" (TOTA). TOTA supports the creation of distributed overlay data structures spread across a distributed network. In this paper, we want to test the TOTA's generality by implementing some advanced mechanisms required for the control of a modular robot. In particular, our goal is to show that TOTA enables to easily and flexibly express and program biologically-inspired control algorithms (i.e. hormone-based [3]) for modular robots' coordination.

2. THE TUPLES ON THE AIR APPROACH

TOTA is a general-purpose middleware for multiagent coordination, in distributed computing scenarios. In TOTA we assume the presence of a network of possibly mobile nodes, each running an agent application. Each agent is supported by a local version of the TOTA middleware. Nodes are connected by only short-range network links. Each agent has only a local (one-hop) perception of its environment. Long range interactions must be mediated by other agents. Upon the distributed space identified by the dynamic network of TOTA nodes, each agent is capable of locally storing tuples

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

and letting them diffuse through the network. Tuples are injected in the system from a particular node, and spread hop-by-hop accordingly to a specified propagation rule (see Fig. 1(left)). In the modular robot scenario, this reflects in having each agent installed in a module of the robot. The modules (and thus the agents) are connected according to the topology of the robot. Tuples are injected and propagate across the robot's body (see Fig. 1(right)).

In TOTA, distributed tuples $\mathbf{T}=(\mathbf{C},\mathbf{P},\mathbf{M})$ are characterized by a content $\mathbf{C}$, a propagation rule $\mathbf{P}$ and a maintenance rule $\mathbf{M}$. The content $\mathbf{C}$ is an ordered set of typed fields representing the information carried on by the tuple. The propagation rule $\mathbf{P}$ determines how the tuple should be distributed and propagated in the network, and how the tuple content should change during propagation. The maintenance rule $\mathbf{M}$ determines how a tuple's distributed structure should react to events occurring in the environment.

We developed a first prototype of TOTA running on Linux IPaQs equipped with 802.11b WLAN and Java (J2ME, CDC, Personal profile). Moreover, we have implemented an emulator to analyze TOTA behavior in presence of hundreds of nodes [1]. To perform experiments with modular robots, we connected our TOTA simulator with the Polybot modular robot simulator [2]. This program can simulate the behavior of various types of modular robots, taking into account both the characteristics of the joints connecting different parts of the robot and the physical forces acting on the robot (e.g., gravitation), and offers a 3D view of the robot's actual configuration and movements.

3. MULTIAGENT COORDINATION IN MODULAR ROBOTS

From a multiagent perspective, the main challenge in modular robots is the design of decentralized coordination mechanisms enabling autonomous agents (i.e., modules) to coordinate their actions to achieve a specific global shape, or a specific motion gait. Some of the most innovative approaches to control a modular robot adopt the biologically inspired idea of hormones [3]. Hormone signals are actually sort of messages, spread across the robot and triggering the individual actuator's bending. A motion gait would be encoded by means of a specific sequence of hormones to be injected in the robot and by means of specific reactions triggered by these hormones, changing the bending angles.

The idea of hormones is a perfect match for TOTA distributed tuples and our approach has been to re-implement hormones with the support of our middleware. With this regard, it is fair to remark that we do not propose a novel approach for modular robot coordination. We just take advantage of the TOTA middleware to implement (with some changes) the hormone-based approach [3]. In our opinion, the importance of how to actually program such systems should not be overlooked and the proposed approach is valuable with regard to this topic.

In particular, the main subject of our research has been chain-type modular robots. This kind of robots is characterized by the fact modules are connected in a line configuration (e.g., snake-like), or - eventually - in a tree-like configuration (e.g., robot with legs), see the figures in next pages. In our experiments, we assumed that the robot is composed of very simple equal modules (i.e., joint actuators). Each module has a "front" side and a "back" side. On each side

there is an infrared (IR) network link. Each module runs the TOTA middleware and an agent in charge of driving the module joint. The agent, looking at the active IR links, is able to infer whether it is the "head," the "tail," or a part of the "body" of the robot. Specifically, the "head" agent is the one with only the back IR link active, the "tail" agent is the one with only the front IR link active, a "body" agent is one having both the IR links active. The process of assessing whether an IR link is active or not can be based on "ping" messages and can be executed iteratively to take into account topological reconfigurations and module breakdown.

From a methodology point of view and in very general terms, the proposed approach consists in codifying a motion gait by means of a *Gait* tuple (examples are in the next section). Once this tuple is injected in the modular robot, it propagates hop by hop across all the modules changing its content to the desired gait. When an agent installed on a module senses the income of a tuple of this kind, it reacts by bending the module joint by the angle specified in the tuple. So, for example, if in a robot composed of N modules a tuple having in its content a fixed angle of about $(360/N)^\circ$ is spread, the robot closes into a loop.

4. EXPERIMENTS

In the rest of this section we will use TOTA to support three motion gaits in modular robot: the "caterpillar gait" (that lets the robot proceed by mimicking the motion of a snake) and the "rolling gait" (that lets the robot close in a loop, and then roll). Finally, we will present a gait for legged robots where agents coordinate legs' movements to proceed forward. It is worth noting that while the mechanisms underlying the former two gaits are well known in modular robot research [3], the latter one has been designed from scratch.

4.1 Caterpillar Gait

To implement the caterpillar gait, the head agent injects a caterpillar tuple (i.e., a TOTA tuple of the class *Caterpillar Gait Tuple*). The general structure of such a tuple is depicted below it propagates across the robot letting it bend accordingly. Once the tail agent receives the tuple, according to the general description given above, it injects another tuple to notify the head that a new step is ready to be executed. At this point, the head agent repropagates the tuple. This process is iterated letting the whole robot move performing the caterpillar gait (see Fig. 2).

Caterpillar Gait Tuple

$\mathbf{C} = (\text{state}, \text{angle})$

$\mathbf{P} = (\text{propagate hop-by-hop, storing on intermediate nodes changing the content accordingly to the following table})$

Current State	New State	New Angle
INIT	A	$+45^\circ$
A	B	$+45^\circ$
B	C	-45°
C	D	-45°
D	A	$+45^\circ$

$\mathbf{M} = (\text{If on the head node and upon the receipt of a gait-tuple, re-apply propagation})$

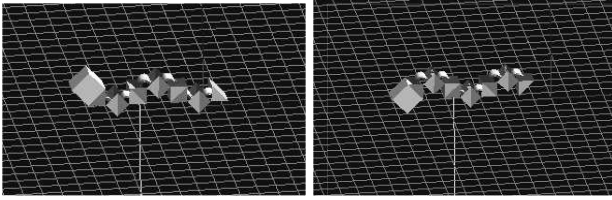


Figure 2: Some stages of a caterpillar gait, in a chain-typed modular robot, composed of six actuators

4.2 Rolling Gait

The idea of this gait is to let the robot close in a loop and then roll. Unfortunately, the modular robot simulator we employed does not allow structures with loops. Structures with loops are overconstrained. The simulation does not solve the constraint satisfaction problem. To overcome this problem, we let the robot bend in an open loop (something like a ‘C’ shape) and then roll. Although this complicates the rolling procedure, it allows us to maintain the general approach described before. In fact, we still have a “head” and a “tail” agent that would be otherwise removed if the loop were actually closed (i.e., with only “body” agents). The *Rolling Gait Tuple* is the tuple employed to let the robot roll. In general terms, it can have two states, T (turn) and F (flat). Consider, for example, a robot composed of 12 modules and assuming a turning angle of 45° . A tuple spread in the robot with a distributed shape like “FTTFTTFT-TFTT” closes in a loop. Then, if the tuple changes its content by “rolling” the above string (like the ROL assembler command), the robot performs the rolling gait. From the single tuple point of view, this consists in changing its content to assume values F - T - T iteratively. It is worth noting that such a kind of content change critically depends on the number of modules composing the robot and the number of turns we want to implement to let the robot close in a loop. For example, three turns of 60° each create a triangular track, four turns of 90° each create a rectangular track, etc. Moreover, it depends on the number of modules involved in each turn. Despite all these parameters, it is rather easy to build a general algorithm enabling a tuple to create dynamically, at runtime, the sequence of F and T states it has to cycle and the corresponding turning angle to enable the rolling gait. Some snapshots showing the rolling gait in action are in Fig. 3.

Rolling Gait Tuple

C = (state, angle)

P = (propagate hop-by-hop, cycling between the states F - T - T. Set the angle to 45deg if the state is T. Set the angle to 0 if the state is F)

M = (if the network topology changes, restart propagation according to new head, body and tail position)

4.3 Walking Gait

We performed this experiment on a “legged” robot that can move by coordinating legs movements. The idea of this experiment is to have the modules of the robot connected in a 6-legs configuration, and then coordinate the actions of these modules so as to let the legged robot walk. The robot

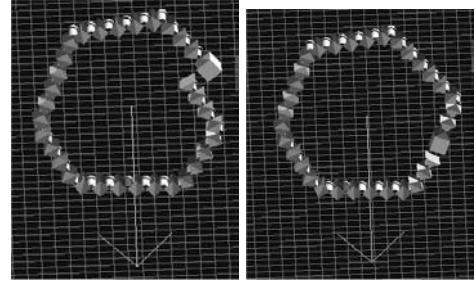


Figure 3: Some stages of a rolling gait in a chain-typed modular robot composed of 32 actuators.

can start moving the legs to proceed upward. This again is realized by letting the head of the robot inject another tuple that propagates across the modules. This *Walker Gait Tuple* tuple is very simple: it alternatively lets the left and right robot legs swing 45° forward. Some screen-shots of the actual robot movement are presented in Fig. 4.

Walker Gait Tuple

C = (state = LEFT—RIGHT, angle)

P = (propagate hop-by-hop. if current node is a spine node switch state. if state == LEFT turn left legs to 45deg and right legs to -45deg . if state == RIGHT turn right legs to 45deg and left legs to -45deg .)

M = (if the network topology changes, restart propagation according to new head, body and tail position)

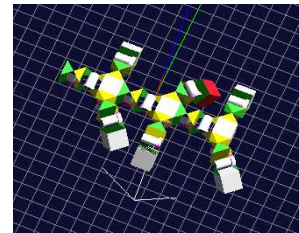


Figure 4: The legged robot walks by coordinating legs movements.

5. ACKNOWLEDGMENTS

Work supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

6. REFERENCES

- [1] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the tota middleware. In *Proceedings of the International Conference On Pervasive Computing (Percom)*. IEEE CS Press, Orlando, Florida, USA, 2004.
- [2] Modular reconfigurable robotics at parc. <http://www2.parc.com/spl/projects/modrobots>.
- [3] W. Shen, B. Salemi, and P. Will. Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *IEEE Transactions on Robotics and Automation*, 18(5):1 – 12, 2002.