A Distributed Coordination System for Modular Reconfigurable Robots

Zhicheng Deng
Nokia Research Center
Nokia (China) Investment CO.,LTD.
Beijing, China
ext-zhicheng.deng@nokia.com

Wei Wang
Robotics Institute
Beihang University
Beijing, China
wangweilab@buaa.edu.cn

***Abstract*—Reconfigurable robotic systems are made from a large number of independent modules that can be used to form various robot shapes. Consequently the robots have the capabilities to optimize their configurations to suit different tasks and environments. This reconfigurability needs not only appropriate module structures, but also corresponding communication and control systems to coordinate the modules' behavior. This paper presents a distributed coordination system that supports multi-point to multi-point communication and combines a set of distributed algorithms for accomplishing global tasks in a modular reconfigurable robot. The system allows discovering the number of robotic modules dynamically, solving task conflicts by integrating priority scheduling and FIFO algorithm, collaborating modules' behaviors to fulfill a designated task, and achieving distributed behavior synchronization.**

Keywords-coordination; modular; reconfigurable; distributed system; robot

I. Introduction

Recently, a lot of work has been done on reconfigurable robots composed of several modules which are able to optimize their configurations to adapt to different tasks and environments[1]-[5]. For instance, a multi-module robot-PolyBot adopts rolling type for passing over flat terrain, earthworm type to move in a narrow space and a spider type to stride over uncertain hilly terrain [6]. This reconfigurability is very essential in tasks such as urban search, reconnaissance, rescue, transportation and maintenance where robots must confront unstructured environments and carry out tasks that are difficult for a fixed-shape robot [7].

The application of modular concept to robots will make them more versatile, robust and efficient in different environments. To take full advantage of the modular reconfigurable robots, one must know how to coordinate every module's behavior. However, controlling such systems are very difficult stems from the following facts[8]:

1) The number of the robotic modules is dynamic. New robotic modules may join the system, and existing modules may fail or disconnect from the communication network at any time.

2) Each module has limited resources and does not have the capabilities to perform tasks individually. Generally speaking, the collaboration is the only way for the robot to implement complicated tasks.

3) Different module should have different behavior to fulfill a designated task.

4) Distributed modules execute behaviors asynchronously.

By far, there are two methods for coordinating modular reconfigurable robots: centralized methods and distributed methods.

Centralized methods include a table look-up method used by Yim et al in PolyPod robot [9], a two-layer motion planning method developed by Yoshida et al in M-TRAN robot [10]. Compared with distributed methods, the centralized systems have simple control structures, but they lose a lot of advantages such as robustness, fault tolerance, scalability, versatility and etc.

The advantages of distributed control methods have attracted many researchers and many systems have been developed, which include a distributed control system for lattice-type modular robots developed by Murata et al[11], a MAS based control system proposed by Bojinov et al in Porteo robot[12], a Hormone-based distributed system developed by CONRO research group [13], a self-organizing metamorphic strategy proposed by Xu wei et al [14]. The distributed methods mentioned above assume that the robotic modules can only communicate with their adjacent modules. However, some of modular robotic systems adopt broadcast communication systems. Obviously, these methods are not very suitable for this kind of communication systems. Furthermore, their task coordination mechanisms do not provide ways to coordinate modules' tasks in terms of task importance.

This paper is intended to develop a distributed coordination system for modular reconfigurable robots to meet the requirements of modular, flexible and scalable hardware structure. It is necessary to solve the four problems caused by modular structure. Based on this consideration, a distributed coordination system is developed, which utilizes WLAN 802.11b communication networks, supports multi-point to multi-point data communication. This system combines a set of distributed algorithms for accomplishing global tasks in a modular self-reconfigurable robot. The system allows discovering the number of robotic modules dynamically,

985-Project of MOE, China

DOI 10.4108/ICST.WIOPT2008.3165

solving task conflicts by integrating priority scheduling and FIFO algorithm, collaborating modules' behaviors to execute a designated task, and synchronizing module's distributed behaviors.

II. Discover Robotic Module Dynamically

Dynamic changes including adding or deleting modules in a modular system could happen when the robot is performing tasks. Hence, the robot should be able to detect these dynamic changes and generate a new behavior plan to execute the designated task.

As shown in Fig. 1, a modular reconfigurable robot with WLAN communication network includes several modules R_i (i=1,2,…,n) and several bridged network APj (j=1,2,…,m).

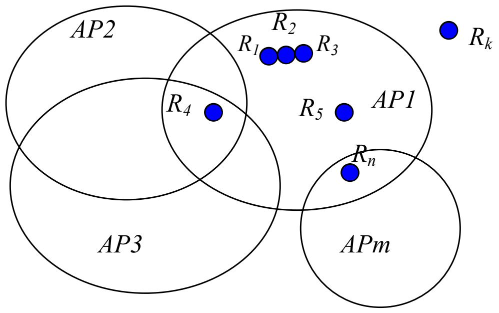


Figure 1. a modular system with WLAN communication network.

As for such a system, we assume that the bridged network uses DHCP to assign IP addresses to the modules connected to the network. (The IP address ranges from x.x.x.1 to x.x.x.254) Therefore, the robotic modules R_i (i=1,2,…,n) in Figure 1 could obtain IP addresses as follows:

$$R_1\text{:IP1}\quad R_2\text{:IP2}\quad R_3\text{:IP3} \ldots R_{k-1}\text{:IPk-1}$$
$$R_{k+1}\text{:IPk+1}\ldots R_n\text{:IPn}$$

After the modules acquire IP addresses from the DHCP server, they will begin to broadcast "Module_Exist" (ME) messages in the net to indicate their existence, the format of the message is:

ME:	Exist	Module's IP address

Every module in the net received the "ME" message will derive the IP address from it and set the corresponding bit in the "Module_Existence_Table" (MET, the format as show in Fig. 2) to "1".

MET:	1	2	…	254

Figure 2. MET structure.

By this means, the module in the network could receive all the "ME" messages from the other modules in the same broadcast domain. Also each module could get the number of the modules that are able to execute the task cooperatively by counting the number of "1" in "MET". Moreover, in order to detect the changes of the robotic modules number dynamically, the bits in the "MET" would be cleared to "0" when "ME" messages are not received from corresponding IP addresses in a specified time interval. Figure 3 illustrates this procedure.

```
//sub-route 1#
while(1){
    if(connected to the network)
            broadcast ME;
    wait(seconds)
}

//sub-route 2#
while(1){
    Exist_delayi++;          //i=0,1,…255;
    Wait(seconds);
    if(Exist_delayi>specified value)  //i=0,1,…255;
    {
          Write "0" to the ith bit in MET;
          Exist_delayi=0;
        }
}

//sub-route 3#
when(received ME){
    derive the IP address from the ME;
    set the corresponding bit in the MET to "1";
    the Exist_delay variable related this IP is cleared to "0";
    Count the number of "1" in MET;
}
```

Figure 3. The procedure of discovering robotic module dynamically.

III. Task Priority Based Task Scheduling

Modules of a modular reconfigurable robot are standalone systems. They do not transfer all the information to other modules. Hence, it is possible for each module to initiate different tasks according to its own sensors. However, a single module with limited resources is not able to perform a complicated task in most cases. In addition, the tasks initiated by different modules are conflicting in some situations. Therefore, it is necessary to set up a task scheduling mechanism that could make all the modules select the highest priority task from all the initiated tasks or select the first task when the tasks have the same priority.

A. *Task negotiation mechanism*

According to the task negotiation requirements, An N-dimension variable vector "Priority" representing tasks importance levels and N FIFO queues "Task FIFO" including n x m task entries are created in each module. The structure of the variables is shown in Figure 4, the FIFO queue j (j=1,2,…,n) stores tasks with the same priority j, and treated by the task manipulation system in terms of first in first out principle.

Priority:	0	1	…	N-1

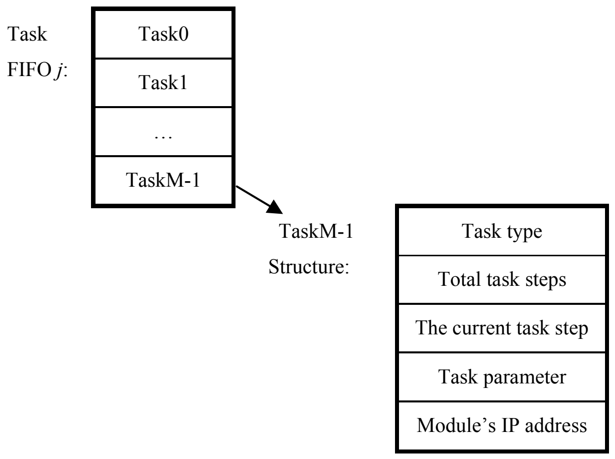


Figure 4. Variables and data structures.

When a task is initiated, a priority will be assigned according to the task importance. ("N-1" means the lowest priority, "0" means the highest priority.) Then, the task initiation module will broadcast a "Task_Negotiation" (TN) message which contains task type, task priority "*j*" and module's IP address information in the net, set the *j*th bit in the vector Priority to "1", add this task to the Task FIFO *j* and create a "Task_Ack_Table" (TAT) variable for this task. The "TAT" variable is used to indicate ACK status from the other modules. Initially, all the value of the table elements are "0", that means no task is to be executed at any priority level. Figure 5 shows the structure of the "TN" and "TAT".

TN:

Task *i*	Priority *j*	Module's IP address

TAT:

1	2	…	254

Figure 5. The structure of TAT

The other modules in the net will also set the *j*th bit of the Priority variable to "1" and add the task to the corresponding FIFO queue when they receive "TN" message. Afterwards, they check the executing task priority (*Pcurrent*) and the highest priority (*Phighest*) in the Priority variable. If *Pcurrent*<= *Phighest*, the module keeps performing the current task, while if *Pcurrent*>*Phighest*, the module stores the current task step, suspend the current task, select the task in the Task FIFO$_{Phighest}$ and broadcast "Task_ACK" (TA) message in the net. The data structure of "TA" is:

TA:

Task_Ack	IP address of the task initiation module	Module's IP address

In addition, when the number of the robotic modules is changed or the current task is finished, the task negotiation mechanism will be triggered as well for all the modules to select one task with the highest priority in the FIFO queues.

B. Termination of the task negotiation

After receiving "TA" message from the other module, the task initiation module will derive the second part (IP address of the task initiation module) from the message, and compared it with its own IP address, if they are the same, the bit of "TAT" corresponding to the third part of the message (Module's IP address) will be set to "1". Once the "TAT" is equal to "MET", the task negotiation process successfully terminates, which means all the modules agree on to execute this task.

Figure 6 illustrates the procedures of the task scheduling method.

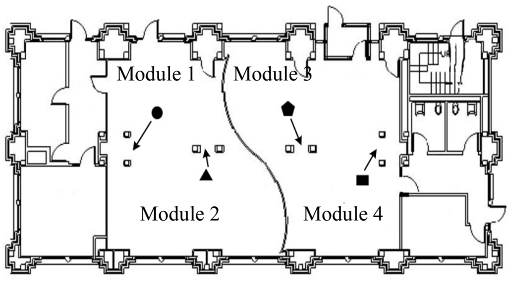
```
// Task initiation (running in the task initiation module)
Task_initiation(){
        …
        Initiate a task based sensor and robot status;
        Assign task priority j;
        Broadcast TN message;
        Set the jth bit of the Priority to "1";
        Add the task to FIFOj queue;
        Create a TAT variable;
        Set the bit of TAT corresponding to module's IP address to "1";
}

//Task negotiation (running in the other modules)
While(1){
        When (received TN message or the current task is finished){
             Set the jth bit of the priority to "1";
             Add the task to FIFOj queue;
              If(Pcurrent>Phighest){
                  Stores the current task step;
                  Select the task in the Task FIFO_Phighest;
                  Broadcast TA messages;
              }
             Else{
                  Keep performing the current task;
              }
        }
}
```

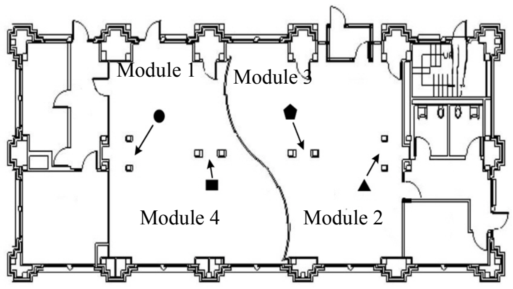
Figure 6. Task scheduling procedure.

IV. Pose and Position based Behavior Selection

It is important that the modules collaborate to execute a set of appropriate local behaviors which utilize the modules' resources in way that the whole effect of all the modules' efforts can accomplish the selected task. As mentioned before, the pose and position of modules varies dynamically. Therefore, the behavior of each module should be selected on the basis of its pose and position relations with the other modules. Figure 7 illustrates a situation wherein module 2 is identical with module 4; the positions of module 2 and module 4 in figure 7 (a) are different from those of in figure 7 (b). Thus, module 2 and module 4 in Figure 7 (b) have to select different behaviors from those of in Figure 7 (a) to fulfill the same global task more effectively.



(a)



(b)

Figure 7. Execution of the task with different modules' position relations. (a) 4 modules execute a designated task. (b) 4 modules with different initial position relations execute the same task in (a).

In order to make all the modules to select the right behavior, a distributed approach is proposed base on the assumption as follows: A global task can be divided into many behaviors relevant to modules' pose and position relations, and the modules can accomplish the task by coordinating the execution of the behaviors in a right way. This assumption can be expressed as:

$$T = \sum_{i=1}^{n} behavior_i$$

Where, T is the designated task; $behavior_i$ is the ith behavior; n represents the number of behaviors that the task can be divided into. (n varies depending on tasks).

After dividing the task into many behaviors, each module will inform the other modules its pose and position information via the communication network. In this paper, IEEE 802.11b wireless Ethernet is used for inter-module communication. To simplify the robot location system and make use of the communication network, the software positioning and navigation method using IEEE802.11b indoor positioning technology are introduced to the system to fulfill localization in large office environments. Dead reckoning, ultrasound and IR proximity sensors are only used for short range positioning. As all the modules' pose and position information is obtained, a preset decision rule is adopted for the module to select its own behavior. Figure 8 shows a block diagram of the behavior selection sub-system.

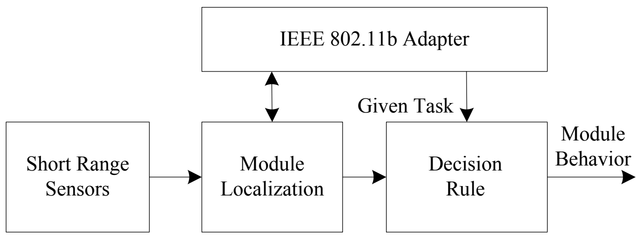


Figure 8. The behavior selection sub-system

V. BEHAVIOR SYNCHRONIZATION

The collaboration of the modules can achieve the valid global effects only if module's selected behaviors are executed in a predefined sequence. In this paper, we proposed to use distributed systems to control the modules. As for such systems, the controllers are independent and modules control processes are asynchronous. This makes the behavior synchronization very difficult, and consequently, a synchronization mechanism is required.

To solve the behavior synchronization problem, we split every behavior to a number of steps. The behavior splitting method is:

1) Different modules' behaviors for one task have the same number of steps.

2) At one moment, modules execute behavior steps with the same sequence number.

3) The asynchronous execution of behavior steps will not affect the global effects.

This behavior-splitting based approach uses execution messages ("Task_Execute", "TE" message) to synchronize the activities of all the modules. Firstly, after the termination of the task negotiation, the task initiation module will broadcast a "TE" message to inform the other modules to start execution of the selected task, and it clears every bit of "TAT". Then, the task initiation module will start the next round of actions when it receives completion messages ("TA" message) from all the other modules (TAT=MET). Finally, after all these behavior steps are executed, the task is successfully accomplished.

TE:	Task type	Behavior step No.	Module's IP address

Figure 9 illustrates the synchronization method.

```
//task initiation module
if(task negotiation terminated){
    behavior step No.=1;
    broadcast TE message;
    clear every bit of  TAT;
    behavior selection;
    execute the behavior step No.1;
    while(behavior step No.<N the number of behavior steps){
            while(TAT==MET);
            behavior step No.++;
            broadcast TE messange;
            execute the behavior step;
            }
}
```

```
//the other modules
While(1){
      if(received TE message){
            derive module's IP address, behavior step No.;
            if(behavior step No.= =1){
                  behavior selection;
                  execute the corresponding behavior element;
                  while(executing the behavior step);
                  broadcast TA message;
                  }
      }
}
```

Figure 9. Modules' behaviors synchronization method.

VI. EXPERIMENTS AND CONCLUSION

A. *Experiments*

Three qualitative experiments have been performed to verify the distributed coordination system on the JL-I robot. JL-I is a modular reconfigurable robot composed by 3 independent modules which can be used to fulfill different tasks. The robotic modules communicate with other modules via wireless Ethernet. Some experiment snapshots are given in Figure 10: a) Adding or deleting a module from the robot JL-I; b) initiating different tasks with different priorities at the same time. (Emergency stop task and moving forward task.) c) Executing structure configuration task. The experimental results showed that the robot can detect the change of the number of robot modules, select a single task among multiple initiated tasks and achieve the global task effects by synchronizing modules' behaviors.

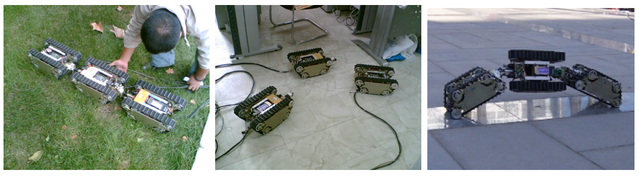


Figure 10. Experiments on modular robot JL-I

B. *Conclusion*

Modular reconfigurable robots have the capability of adopting different configurations to match various tasks. Compared with traditional robotic systems, they are quite suitable for conducting uncertain tasks. However, modularity and reconfigurability also cause problems in task coordination. We proposed a task coordination system for modular robots based on IEEE 802.11b communication network and verified the validation of the system by several simplified experiments. In the future, we will improve the system performance, evaluate the coordination efficiency and try to apply this control system to other non-robotic modular systems.

ACKNOWLEDGMENT

This experiment platform- JL-I is developed under the support from MOE, China. The authors would also give their acknowledgments to Nokia Research Center (Beijing) for technique supports.

REFERENCES

[1] Shigeo Hirose, Takaya Shirasu, and Fumihiko E. Fukuahirna, "A Proposal for Cooperative Robot 'Gunryu' composed of autonomous segments," Robot, Autonomous System, Vol.17, pp.107-118, 1996.

[2] Andres Castano, Alberto Behar, and Peter M. Will, "The Conro modules for reconfigurable robots," IEEE/ASME Transactions on Mechatronics, Vol. 7, No. 4, pp. 403-409, 2002.

[3] Cem Unsal and Pradeep K. Khosla, "Mechatronic design of a modular self-reconfiguring robotic system," Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, pp.1742-1747, Apr. 2000.

[4] Satoshi Murata, Eiichi Yoshida, Akiya, Haruhisa Kurokawa et al, "M-TRAN: Self-Reconfigurable Module Robotic System," IEEE/ASME Transactions on Mechatronics, Vol.7, No.4, pp.431-441, 2002.

[5] Akiya KAMIMURA, Satoshi MURATA, Eiichi YOSHIDA et al, "Self-Reconfigurable Modular Robot-Experiments on Reconfiguration and Locomotion," Proceedings of 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, HI, pp.606-612, Oct.-Nov. 2001.

[6] Mark Yim, David G.Duff and Kimon D.Roufas, "PolyBot: a Module Reconfigurable Robot," Proceedings of the 2000 IEEE International onference on Robotics and Automation, San Francisco, CA, pp. 514-520, 2000.

[7] Castano, A., W.-M. Shen, P. Will, "CONRO: Towards Miniature Self-Sufficient Metamorphic Robots," Autonomous Robots, Vol. 8, No. 3, pp. 309-324, 2000.

[8] Salemi B. Experimental Evaluation of a Distributed Control System for Chain-Type Self-Reconfigurable Robots. Grade School, University of Southern California, 2004.

[9] Yim M.Locomotion with a unit-modular reconfigurable robot, Department of Mechanical Engineering, Stanford University, 1994.

[10] Yoshida E., S. Murata, A. Kamimura et al, "A Motion Planning Method for a Self-Reconfigurable Modular Robot," Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Maui, HI, pp. 590-597, 2001.

[11] Murata S., H. Kurokawa, S. Kokaji, "Self-Assembling Machine," Proceedings of the 1994 IEEE International Conference on Robotics and Automation. San Diego, CA, pp. 441-448, 1994.

[12] Bojinov H., Casal A., and Hogg T., "Multiagent control of self-reconfigurable robots," Proceedings of International Conference on Multiagent System. Boston, MA, pp. 143-150, 2000.

[13] W. M. Shen, Salemi B. and Will P., "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots," IEEE Transactions on Robotics and Automation, Col. 18, No.5, pp.700-712, 2002.

[14] Xu W., Wang S. G., Wang A. L., Yu X. R., "Study on Self-organizing Metamorphosis of Self-reconfigurable Robots," High Technology Letters. No3, pp. 75-78, 2004, in Chinese.