Real-Time Concurrent Task Management for Self-Reconfigurable Robots

Harris Chi Ho Chiu, Wei-Min Shen

Information Science Institute, University of Southern California
4676 Admiralty Way, Marina Del Rey, CA 90292
Email: {chiho, shen}@isi.edu

***Abstract* – We present a task management method for managing computational resources for devices and control of self-reconfigurable robots. This method is useful for controlling multiple devices simultaneously in real-time and makes the development of control software easier. A software framework is developed on top of a real-time scheduler to provide real-time concurrent control over devices and control software. Also, the framework can support a control protocol without global unique identifiers to facilitate the scalability and robustness of reconfigurable robots. We present an implementation and evaluation on a scalable reconfigurable system called Superbot.**

I. Introduction

Self-reconfigurable robots consist of modules having computational units, communication devices, sensors, actuators and connectors. To perform locomotion and reconfiguration [1-2], module components need to be monitored simultaneously in real-time. For controlling a large number of modules, a control protocol without global unique identifiers is essential in handling dynamic and unexpected topology changes [3]. However, conventional single threaded programming methods cannot provide easy management of real-time device responses. Since the accessing time required among devices differs, programmers have to pay extra effort in resolving the timing issue whenever there are new timing changes in communication protocols and device settings. This paper introduces a task management method that provides real-time concurrent control for devices as well as support communication protocols that are free from global unique identifiers.

II. Related Work

Zhang et al.[4] have proposed a software architecture and implemented this on a real-time operating system for modular reconfigurable robots. However, inter-module communication requires a bus and therefore global identifiers are needed for addressing. If a module needs to be added or replaced, the control program needs to be rewritten or another module has to be reprogrammed to retain the same global address. Shen et al. [3] provides control without global addressing by having point-to-point communication between neighboring modules. Program running in every module is identical and can be replicated. Therefore, modules can replace one another. This ensures robustness and scalability. However, the implementation is only single-threaded on CONRO modules. For a more complex reconfigurable robot Superbot [5], control devices have to be controlled simultaneously to demonstrate more advanced locomotion and reconfiguration in real time. It is difficult to be programmed in a single-threaded environment. The control software has to be multi-threaded for concurrent control and also for the ease of writing control software. Our solution is to provide a management method for managing software modules for devices and control software for the ease of control software development and also support global identifiers free control protocol on a reconfigurable robot in a concurrent multithreaded way.

III. Real-Time Concurrent Task Management

A. Reconfigurable Robot and real-Time Scheduler

To perform locomotion and reconfiguration successfully, both control software and hardware control have to meet real-time constraints. For example, a motor needs to be periodically monitored within certain period to hold its position while control software has to send out messages regularly to keep control information updated. However, to manage the computational resources well in real-time, a real-time scheduler with small overhead of memory and processing time is needed to allocate resources among hardware devices and control software properly. At the same time, real-time scheduler should be able to provide software modularity. It provides a common interface for different device software to communicate and also a common mechanism for handling timing constraints. In our implementation on Superbot using Atmega128, AvrX [6] is used as the real-time scheduler and provides tasking, semaphores, timer management and message queues. It fits the requirement of real-time scheduler for reconfigurable robot by having small footprint – about 1500 bytes in code size and 512bytes SRAM for 5 tasks. It takes less than 20ns for Atmega128 at 16 Mhz for context switching.

B. Real-Time Concurrent Task Management Framework

A software framework together with real time scheduler is needed to facilitate control protocol without global unique identifiers. The framework should provide messaging without the need of global addressing and a platform for programmer to develop control software without knowing much detail about underlying system devices.

A thread-like structure called task is adopted from AvrX as a foundation of the framework. In the framework, tasks are divided into system tasks and behavior tasks. A task appears to have a complete CPU control independent of other tasks with its own stack and memory space for static variables. Each task

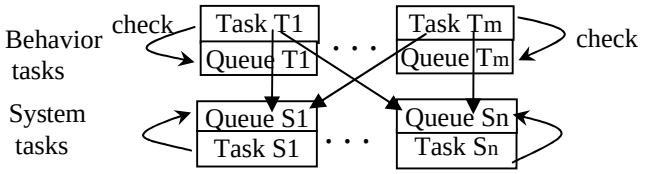

Fig. 1 Tasks interaction—behaviors tasks send messages to system tasks

also has its own message queue for inter-task communication. Tasks send a message by putting the message to the queue of the receiving tasks and receive a message from other tasks by checking its own queue as shown in Figure 1.

System task handles low-level service request for devices. There are 2 tasks for controlling a communication device – sending and receiving task. The sending task listens to its own message queue and only sends when there is message in the queue. For receiving task on a polling communication device, it checks the device at a regular interval for data and generate message for the acquiring task. For interrupt driven devices, received data is collected in the interrupt. A message using dynamic memory buffer is formed and put into the message queue of receiving task. The receiving task picks up messages from the queue and delivers it to destination task. Behavior tasks are responsible to run high-level control software using the functions provided by systems tasks. They have no direct access to the low-level devices and can only call the interfaces provided by the system tasks. This provides an abstraction for programmers by encapsulating system timing details with software interfaces.

C. Supporting Global Identifier Free Control Protocol

To demonstrate the support on global identifier free control protocol, hormone-inspired control [3] is implemented on the framework using Superbot as shown in Figure 2. The control protocol can support locomotion on large configuration such as centipedes. Each module runs same control program. The control program can be easily implemented using the task management framework by introducing 3 behavior tasks – ACSEND, ACRECV and HORMONE task. For Adaptive Communication (AC) protocol, the ACRECV task determines topology information and updates HORMONE task. The ACSEND task is specified for sending probe messages through system tasks for communication interfaces. Each communication interface task checks its queue and uses IR device to send to its neighbors. On the neighboring module, the receiving task of corresponding communication interface will pick up the message and deliver it to ACRECV task. The ACRECV task gets update from its queue and sends corresponding information to HORMONE task. Therefore, the AC protocol (line 1-4) can be carried out.

The HORMONE task checks its message queue for hormone messages and topology information from ACRECV. Upon a hormone message received, HORMONE task can determine what actions to perform and what hormone to propagate to neighboring modules using RULEBASE table. Actions can be executed by putting messages to the queue of task for actuators and the propagation of hormone messages can be achieved by putting message into the message queue of sending task of

ACSEND Task	1 Loop: 2 Send probe messages to all connectors
ACRECV Task	3 Loop: 4 Listen to probe message for topology change
HORMONE Task	5 Loop: 6 For each received hormone message 7 Select and Execute local actions according to module type, local timer, and received hormone data specified in the RULEBASE table 8 Propagate selected hormone message to other connectors through communication interfaces

Fig. 2 Pseudo code for hormone-inspired control

desired communication interface. Therefore, action selection and hormone propagation based on hormone messages and RULEBASE table(line 5-8) can be carried out by just sending message to neighbors. No global addressing on modules demonstrates system scalability on reconfigurable robots.

Upon dynamic changes in topology, ACRECV task either receives or does not receive message in a period of time. It then sends a message to HORMONE task to update its module type. The HORMONE task then selects action and sends out hormone message accordingly. In this way, the robot can still be able to run without reprogramming global address of every module. This facilitates the robustness of reconfigurable robots.

IV. CONCLUSION

This paper presents a real-time concurrent task management method with the use of a real-time scheduler and a software framework on a reconfigurable robot. The real-time scheduler resolves the problem of real-time concurrent device control. The software framework facilitates the scalability and robustness of reconfigurable robots by providing a platform for easy implementation of control protocol without global unique identifiers on each module. It also helps programmer put less effort in understanding the timing constraints of underlying hardware. Additional software peripheral devices can also be added easily.

REFERENCES

[1] Kenneth Payne, Behnam Salemi, Peter Will, and Wei-Min Shen. Sensor-Based Distributed Control for Chain-Typed Self-Reconfiguration. In *Proc. 2004 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pp. 2074–2080, Sendai, Japan, Sept./Oct. 2004.

[2] Wei-Min Shen, Maks Krivokon, Harris Chiu, Jacob Everist, Michael Rubenstein, and Jagadesh Venkatesh. Multimode Locomotion for Reconfigurable Robots. *Autonomous Robots*, 20(2):165–177, 2006.

[3] Wei-Min Shen, Behnam Salemi, and Peter Will. Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots. *IEEE Trans. on Robotics and Automation*, 18(5):700–712, October 2002.

[4] Zhang, Y.; Roufas, K. D.; Yim, M. H. Software architecture for modular self-reconfigurable robots. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001); 2001 October 29-November 3; Maui; HI. Piscataway, NJ: IEEE; 2001; 4:2355-2360.

[5] Behnam Salemi, Mark Moll, and Wei-Min Shen. SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System. In *Proc. 2006 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Beijing, China, October 2006.

[6] L. Barello, *AvrX Real-Time Kernel*. [Online]. Available: http://www.barello.net/avrx