Software Architecture of Contents-based Control for Co-operative Remote Manipulation of Multi-Robots

Dinh Trong Thuy, Soon Ju Kang
School of Electrical Engineering and Computer Science, Kyungpook National University
e-mail: thuydtr@yahoo.com, sjkang@ee.knu.ac.kr

Abstract
In this paper, we propose software architecture for conveying contents-based OpenSound Control (OSC) packet from manipulation user interface to cooperative remote multi-robots. The Flash application is used as a controlling user interface and the physical prototyping of multi-robots were developed using physical prototyping toolkit.

1. Introduction

Open Sound Control (OSC) [1] is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology and has been used in many applications such as sensor/gesture-based electronic musical instruments, mapping nonmusical data to sound, multiple user shared musical control, web interface and sound synthesis, networked LAN musical performance, virtual reality etc. This simple and powerful control provides several methods needed for real-time control of sound and other media processing. In this paper we propose a software architecture for content based control for co-operative remote manipulation of multi-robots. We believe that the architecture can be applied in number of applications, where systems are located remotely to control using contents such as Flash [2].

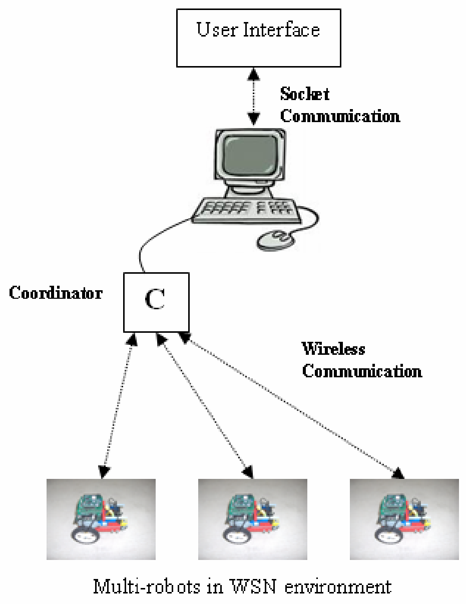
1.1 Motivation

In the recent years the content-based control has been an interesting research area. Few researchers have studied the content based solutions to different domain specific applications. The content based applications are especially useful in environments where the systems can react or response to the data embedded in the contents. These kinds of applications are widely useful in many situations such as sending contents to remotely located multiple co-operative robots to accomplish various goals. With these abilities a user can achieve several missions that can be performed more efficiently and robustly using robots such as working as parts of assembly line, gathering environmental data, oil or energy mining operations. These applications motivated us to control the robots using content-based approach. Our main focus in this paper is on building software architecture for conveying contents-based control message from user interface to cooperative remote multi-robots.

1.2 Proposed Method

While controlling multiple robots it is important that, robots perform in desired ways with real-time connectivity. In many computer applications, it is difficult to identify co-operative robot manipulation remotely without proper control architecture. So, to control such widely spreaded robots in WSN environment as in [3] it is very important to have contents-based control. We have used Flash application as a flexible way to create dynamic contents and to convey its contents to another computer programs.

Figure 1 shows proposed content-based approach for controlling co-operative multi-robots in remotely controlled environment. In our proposed idea Flash is used as user interface that can recognize user manipulations communicating with delivery programs using socket communication. Delivery programs will then transfer control data to a coordinator. Coordinator is responsible to communicate with end-nodes. Each end-node is connected with a mobile robot, which includes number of sensors and actuators to perform desired actions based on control data received from coordinator. We have used ZigBee protocol [4] as a wireless protocol to communicate with multi-robots remotely. ZigBee cab greatly facilitate to directly mapping the node to individual robots.



(Figure 1) Proposed content-based approach for controlling cooperative remotely located multi-robots

Flash application and delivery programs explained earlier can reside in the same computer or in different PCs. The communication between Flash application and delivery programs is socket based communication. The delivery programs include gateway server and OSC client which are mainly responsible for the command deliver to the ZigBee coordinator. If the Flash application and these computer programs run on the same machine, the communication time between them is nearly equal to zero, and thus the communication time between Flash applications with robots depends mainly on communication time between delivery programs and robots. The physical prototyping of robots were developed using LEGO [5] based development toolkit.

2. Related Works

Over the years many researchers developed the applications related to content-based delivery over the network. For example, researchers in [6] proposed the content-based control of goal directed attention during human action perception. Other researchers, as in [7], they developed methods content based retrieval system for medical images. In addition there are several service frameworks such as OSGi [8] and OSC were developed to enhance the flexibility to transfer the multi-media data. The OSGi can be used for dynamic deployment environments and includes higher complexity. Moreover they also have shorter development cycles. It also adds overhead to RAM and flash. The specification process is also complex in OSGi as it involves many participants with different interests. In this paper we aim to create software architecture for communication between Flash and remotely located robots via OSC. We found that OSC is right protocol for our application, because of its communication, multi-media, and ability of real-time processing.

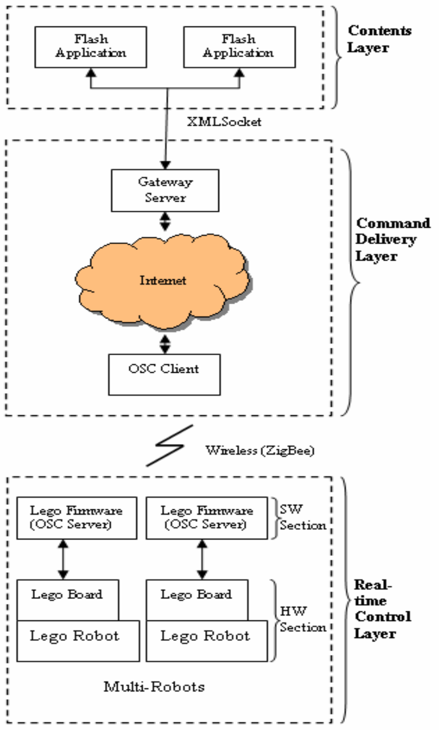
3. Detail Design and Implementation

3.1 Proposed Software Architecture

The overview of the detailed software architecture is shown in Figure 2. The proposed software architecture is a layered architecture with three layers as contents layer, command deliver layer, and real-time control layer. Contents layer includes Flash applications that act as user interfaces. Multiple Flash Applications can control the robots' activities concurrently. Flash application will encode OSC data (Flash's contents or Flash's dynamic data) into XML data type before transferring it to command delivery layer.

Command delivery layer includes gateway server module and OSC client module. Gateway server acts as gateway between TCP world of Flash XMLSockets and UDP world of OSC. Gateway server receives XML-encoded OSC data from Flash application via XML socket, parses XML-encoded OSC data, and then sends parsed OSC data to OSC client via the Internet. OSC client then encode OSC data received from gateway server into ZigBee-encoded OSC data. OSC client communicates with LEGO firmware via ZigBee.

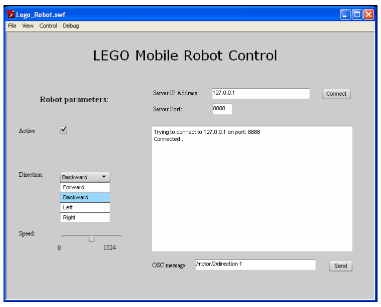
Real-time control layer comprises both software (SW) section and hardware (HW) section. Software section includes the LEGO firmware which acts as OSC server that is embedded on LEGO hardware board. Each LEGO hardware board is connected to a LEGO robot which contains several motors and sensors. Firmware on LEGO board will control motors and sensors, and thus, control robot's activities.



(Figure 2) Detailed software architecture for content-based control for co-operative remote manipulation of multi-robots system.

3.2 Building Contents with Control File

Flash Player has support for an embedded scripting language called ActionScript (AS), which is based on ECMAScript. Since its inception ActionScript has matured from script syntax without variables to one that supports object-oriented code, and may now be compared in capability to JavaScript. Flash is a common format for games, animations, and GUIs embedded into web pages [9]. Our Flash application is programmed using Adobe Flash CS3 Professional authoring tool with ActionScript 3.0. Figure 3 shows the user interface that we used in our experiment. User needs to setup connection between content layer and command delivery layer using server IP address and port parameters. Once the connection between Flash and LEGO board is established, user can control robots' activities using dynamic data (OSC message typed in *textInput*) or using contents (checkBox, comboBox, and slider that indicate robots' active status, robots' direction, and robots' speed respectively).



Server IP Address: 127.0.0.1
Server Port: 8888
Active ✓
Direction: Backward

```
Trying to connect to 127.0.0.1 on port: 8888
Connected...
```

OSC message: /motor/2/direction 1

(Figure 3) An example of Flash application used in our implementation

3.3 OSC Data Format

The unit of transmission of OSC is an *OSC Packet*. Any application that sends OSC packets is an *OSC Client*; any application that receives OSC packets is an *OSC Server.* An OSC packet consists of its *contents*, a contiguous block of binary data, and its *size*, the number of bytes that comprise the contents. The size of an OSC packet is always a multiple of 4.

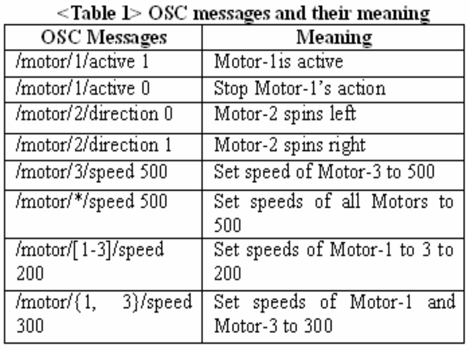
The underlying network that delivers an OSC packet is responsible for delivering both the contents and the size to the OSC application. An OSC packet can be naturally represented by a datagram by a network protocol such as UDP. The contents of an OSC packet must be either an *OSC Message* or an *OSC Bundle*. The first byte of the packet's contents unambiguously distinguishes between these two alternatives. In next sections, we will describe OSC data type using in our implementation.

3.3.1 OSC Message

OSC message is based on the notion of '\b' messages, which are composed of an address, and the data to be sent to that address. The address looks a lot like a URL that we may type into internet browser. Each element in the address is like a directory, with other elements inside it, and each element is separated from the next by a slash (/). Any number of '\b' argument '\b' values can be sent to that address by including them in the message after the address. These values can be integers or floating point numbers. Address and data are separated by a space.

In our implementation, we have physically built robots which include both sensors and motors. Each motor has three properties: *active, direction*, and *speed*. Active, direction and speed all are integers. Active indicate running status of the motor and has 2 values. Value 1 for running the motors and 0 for stopping it. Direction has 2 values: 0 or 1. If the direction value is equal to 0, the motor spins to the left. If the direction value is equal to 1, the motor spins to the right. Speed value of the motor is on the range from 0 to 1024. Speed value is equal to 0 means motor is not running. Higher value of speed means faster spinning of motor. Possible OSC messages that are used in our implementation are shown in Table 1.

<Table 1> OSC messages and their meaning



OSC Messages	Meaning
/motor/1/active 1	Motor-1is active
/motor/1/active 0	Stop Motor-1's action
/motor/2/direction 0	Motor-2 spins left
/motor/2/direction 1	Motor-2 spins right
/motor/3/speed 500	Set speed of Motor-3 to 500
/motor/*/speed 500	Set speeds of all Motors to 500
/motor/[1-3]/speed 200	Set speeds of Motor-1 to 3 to 200
/motor/{1, 3}/speed 300	Set speeds of Motor-1 and Motor-3 to 300

3.3.2 OSC Bundle

An OSC bundle consists of the OSC-string "#bundle" followed by zero or more *OSC Bundle Elements*. An OSC bundle element consists of its size and its contents. The size is an *int32* representing the number of bytes in the contents, and will always be a multiples of 4. Definition of bundle is recusive definition. It means, bundle may contain bundles. Example of OSC bundle that is used in our implementation is shown in Figure 4.

#bundle			Bundle Data
20			Size of first element
/motor/2/active		1	Enclosed message
68			Size of second element
#bundle			Enclosed subbundle
24	/motor/2/direction	1	Messages in subbundle
20	/motor/2/speed	500	

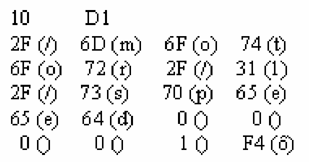
(Figure 4) Example of an OSC bundle

3.3.3 XML-encoded OSC

Flash will encode OSC data (Flash's contents and Flash's dynamic data) into XML-encoded OSC data type before transferring it to command delivery layer [10]. Suppose we have an OSC message: *"/motor/1/speed 500*". Flash will encode this message into XML-encoded OSC data as follows:
"<OSCPACKET ADDRESS="127.0.0.1" PORT="8888"><MESSAGE NAME="/motor/1/speed"><ARGUMENT VALUE="500" /></MESSAGE></OSCPACKET>"
Parameter: ADDRESS="127.0.0.1" is IP address of computer that runs Gateway Server. Communication between Flash application and gateway server is via XMLSocket on Gateway Server port: 8888.

3.3.4 ZigBee-encoded OSC

Gateway server receives XML-encoded OSC data from Flash application, and then parses this data. Parsed OSC data is transferred to OSC client. OSC client will encode OSC data that received from gateway server into ZigBee-encoded OSC data, which contents a ZigBee end node address and hexadecimal values of bytes of OSC data. For OSC message: "/motor/1/speed 500" above, ZigBee-encoded OSC data will look like as below:



```
10      D1
2F (/)  6D (m)  6F (o)  74 (t)
6F (o)  72 (r)  2F (/)  31 (1)
2F (/)  73 (s)  70 (p)  65 (e)
65 (e)  64 (d)   0 ()    0 ()
 0 ()    0 ()    1 ()   F4 (ô)
```

In the example above, 10D1 at the first line is ZigBee end node address, followed by hexadecimal values of OSC message. The corresponding ASCII character in parentheses is not included in actual ZigBee-encoded OSC data. They are shown here only for illustration.

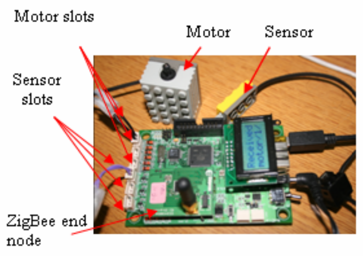
3.4 Communication between Command Delivery and Real-time Control Layer

A ZigBee coordinator is connected with computer that run OSC client module via RS232. Each LEGO board has a ZigBee end-node module that can communicate with ZigBee coordinator described earlier. ZigBee end node receives ZigBee-encoded OSC data from ZigBee coordinator, sends it to LEGO Firmware. LEGO firmware then parses the data, and forces LEGO board's motors perform the corresponding actions. ZigBee coordinator can send one ZigBee-encoded OSC data to many ZigBee end nodes at the nearly same time by changing the ZigBee end node address at the head of transferred data; thus Flash application can control multi-robots that perform same behaviors at the same time.

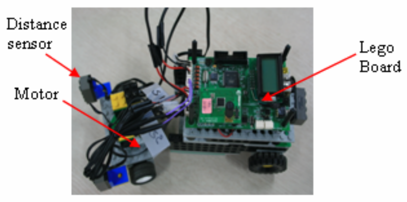
3.5 Implementation

The physical prototypes of robots were developed using LEGO-based kit. The LEGO board is developed in our lab, and is responsible for interacting with sensors and actuators in the robot. In the present prototype, the LEGO board has 3 sensors and 3 motor drive actuators as shown in figure 5. Figure 6 shows the components of the individual robot.

The implementation is performed in a PC with Windows XP operating system environment. Flash application is programmed with ActionScript 3.0 in Adobe Flash CS3 Professional authoring tool. Gateway server is written in Java using NetBeans IDE 6.0 with JDK1.6.0_04. OSC client is developed with C++ programming language under Microsoft Visual Studio 2005, and finally LEGO firmware is developed using Eclipse IDE.



(Figure 5) Implementation Board



(Figure 6) Components of individual robot

4. Experimental Evaluation

In experiment evaluation section, we will measure the transaction time with different sizes of transferring data between Flash application and LEGO Board. The experiment is tested on Windows XP environment and is tested under the environment that has only one coordinator and one ZigBee end node.

Flash application sends OSC packet to LEGO board. When LEGO board receives this OSC packet, it sends this message back to Flash application immediately. Once Flash application receive OSC packet that is sent back from LEGO board, it immediately send next OSC packet. The process above was performed 1000 times, and we measured the average time for one transaction. Overall process described above has been done several times with different sizes of data. We showed the minimum time and maximum time for average time measurement of one transaction in the Table 2 as below:

<Table 2> Average time measurement of one transaction between Flash application and LEGO board

Data size (bytes)	Minimum Time (seconds)	Maximum Time (seconds)
32	0.86	0.87
64	0.87	0.89

5. Conclusion

In this paper we have presented layered software architecture for contents-based control cooperative remote manipulation of multi-robots in network environment. The architecture includes three layers: contents layer, command deliver layer, and real-time control layer. In our approach, we used Flash application as user interface that can convey its contents to LEGO robots, and then can control the robots' activities. The user interface and OSC command delivery modules are resided in one computer. We have used ZigBee to implement communication between command delivery programs to LEGO robots. Therefore it allows us to control of multi-robots performing same activities simultaneously. We also showed the experimental evaluation results of transferring different sizes of data between Flash and LEGO board. Results are found to be satisfactory.

6. Acknowledgements

This research was supported by the Ministry of Knowledge Economy, Korea, under the ITRC support program supervised by the IITA (IITA-2008-C1090-0801-0020).

7. References

[1] OpenSound Control (OSC), http://archive.cnmat.berkeley.edu/OpenSoundControl/
[2] Adobe Flash CS3 Professional, http://www.adobe.com/products/flash/
[3] Laxmisha Rai and Soon Ju Kang, *Intelligent Real-Time Software Architecture Supporting Remote Controlled Group Behavior for Widely Spreaded and Cooperative Mobile Robots with Sensor Network*, Proceedings of 13th IEEE Intl. Conf. on Embedded and Real Time Computing and Applications,p.363-368, 2007.
[4] ZigBee Alliance, www.zigbee.org
[5] LEGO Home Page, www.legomindstorms.com
[6] Yiannis Demiris, and Bassam Khadhouri, *Content-based control of goal-directed attention during human action perception*, The 15th IEEE Intl. Symp. on Robot and Human Interactive Comm., pp. 226-231 , 2006.
[7] Tatjana Zrimec, *A content-based retrieval system for medical images*, 7th Intl. Conf. on Control, Automation, Robotics and Vision, pp 180-185, 2002.
[8] OSGi alliance, www.osgi.org
[9] Adobe DreamWeaver http://www.freewebs.com/adobedreamweaver/
[10] Adrian Freed, Andy Schmeder, Michael Zbyszynski, *Open Sound Control, A flexible protocol for sensor networking*, http://opensoundcontrol.org/files/OSC-Demo.pdf