An Efficient Data Sharing Solution of Multi-Functional Robotics Design: QFlow

Li-Wei Lu , N. Michael Mayer
Department of Electronic Engineering,
National Chung Cheng University, Taiwan R.O.C

Abstract

In modern robotics structure design, robot usually composed with several functioning units or mechanical parts. Take service robot as an example, a service robot may contain a vision module, a arm controlling module and reception module etc.. One could easily deduces information/data sharing between modules is an important issue. Hence, we propose a program called *QFlow*, *QFlow* helps the developers focus on the module design, the information sharing is realized by *QFlow* with specifying plugs and modules inside *QFlow*, and information is transfer under UDP-protocol which handling by the server of *QFlow*.

1 Introduction

In a multi-functional robotics design, robots often capable with many functionalities, mapping and localization, vision processing, voice recognition and robot arm controlling etc., which require a better collaboration within every part of the design. Modulization is becoming a useful technique which divided a large-scale program into several functioning modules.

Various methods have been proposed to solve innermodule communication. [6] implemented a responsive processor as a platform for parallel/distributed control and the communication among *Responsive Processors* is handled by the *Responsive Link* communication standard. [7] proposed and implemented a dual bus architecture and the dynamic switching of the information flow structure by the modules themselves. However, we instead choose the wireless communication as a medium which give us a great advantage on integrated our robot system easily.

In our service robot project, we modulize every functionality as a independent module. We now depict a situation for a better understanding. Imagine a service robot receiving the *"GOTO"* voice command from the user, the audio stream then processed by the voice recognition module which analyze the the location demanded

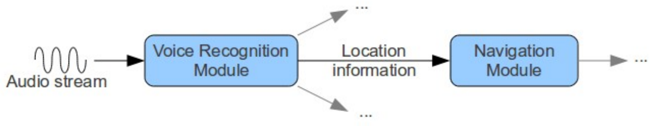

Figure 1: A simple example that voice recognition receive and analyze a audio stream then pass to another module, says, navigation module

and pass to the navigation module (Fig. 1). This information/data sharing behavior could happened massively. Hence we propose *QFlow.*

QFlow is a program that helps the developers focus on the module design, the developers only have to specified the inputs and outputs of the module. Meanwhile, *QFlow* will treat every module as a black box process, it only concerns the collection of information and delivering it to the correct modules using the UDP-protocol.

In the next section, we will talk about our motivation. Then, we will describe our the backbone structure of *QFlow* and some basic functioning unit such as Plug Unit, Behavior Unit and Connector Unit etc..(Section 3). Later, we explain the essential part of *QFlow*, Action Server, and the XML synchronization in section 4.

2 Motivation

Currently, we compete at the RoboCup@home [2] competition. During the competition, there will be some potential chances we need to modify a individual module on the fly. In order to achieve better integratability, we use wireless communication as medium under UDP-protocol. Furthermore, *QFlow* allows every module which in a form of dynamic loadable library doesn't have to exist on the same computer, in the mean time, *QFlow* will handle the communication among modules. The approach allows that multiple developers easily contribute code on the specific modules and can integrate the system effortlessly.

3 Backbone structure of *QFlow*

QFlow is a higher level server that handling the structure of several robot behaviors. In our robotics design, we try to built every functionality into a dynamic loadable library which is able to load by *QFlow* and encapsulated to Behavior Unit. Our module loader can load and unload dynamic modules, and the loader is implemented based on the singleton design pattern which ensure our module loader class has only one instance at all times.

In the following, we will further introduce every function unit we employed in detailed.

3.1 Functioning Unit

3.1.1 Behavior Unit

This unit is basically the module itself, he behavior unit can be view as a blackbox process. As a module, behavior unit also has inward and outward data flow, in fact, the developers of the modules should assign those information inside their code block in a certain form (See Fig. 2 a.). Once the *QFlow* add a module in, the constructor of dynamic module is trigger, each module will do it's task independently, however, data sharing is not functioning at this stage until *QFlow* trigger the "start event". The "start event" is a signal telling *QFlow* to start working on data sharing process, in other words, which means every module starts to receiving and sending data. (See Fig. 2 b.)

3.1.2 Plug Unit

Plug unit is a visualized representation of inward and outward data. This information comes from the module, the developers of modules should assign the amount of inwarding and outwarding data flow, and can surely give a name on it. While *QFlow* loading in the module, it analyzes the modules and attaches the Plugs unit to Behavior units. Note that one is not able to modify any plugs attached to behavior units, it can only be specified by the developers of modules. A plug itself doesn't know who it's data is forwarding to nor the data sender.

3.1.3 Connector Unit

Connector unit is a visualized representation of data flow, there are two plug units on both end (See Fig. 3). Furthermore, The direction of those two plug must be reverse, that is, one input and one output plug. This unit is important for *QFlow*, it indicates *QFlow* where the message is forwarding to, we will go into it in more detail in next section.

3.1.4 Net Unit

One key feature of our approach is that we let the data transferring via wireless medium under UDP-protocol. Hence, specifying network information is necessary, we employ Net Unit that one can specify both IP address and port number. The Net Unit is strictly combined with the Connector Unit since Connector Unit represent where the data flow comes from and where it goes. (See Fig. 4)

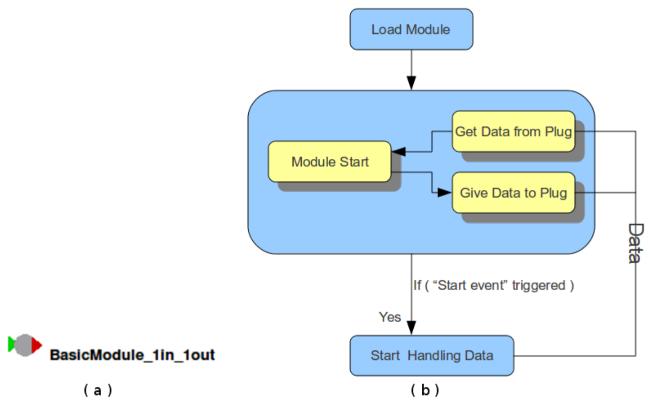

Figure 2: a) The Behavior Unit. The green triangle means input direction, and the red one means output direction. b) The flow chart shows the procedure how *QFlow* handling behavior unit. The data sharing is proceeding only if the "Start Event" is triggered.

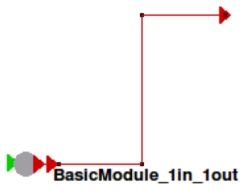

Figure 3: A Connector Unit. The unit is the visualization of data flow, a net unit is able to be attached on it.

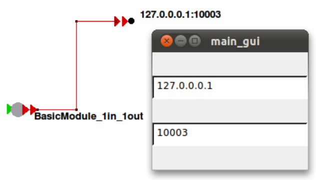

Figure 4: A Net Unit is attached on the end of a Connector Unit. In this example, *QFlow* sends the appropriate message to a IP 127.0.0.1 on 10003.

4 Action Server

Before we get further, we would like to reiterate that our purpose of *QFlow* is to provide a fairly clear approach of existing severals module when implementing a robotic system, in the mean while realize the data sharing ability within modules. Thus, *QFlow* is a higher level server that handling the modules of robotic behaviors in a structured way.

Before the "Start Event" of data sharing, Action Server will register all functioning units and creates a list of it. During the sharing stage, Action Server provides a routine running all the time, it checks whether the memory space of each Plug Unit has updated and determine the location in which the data should be. If the destination is on the same computer(determine by IP address), action server simply copy the memory space, on the contrary, action server will transfer the data under UDP-protocol if the destination is on the the other terminal.

QFlow handles all the data communication within modules, one could say that Action Server is the essential part of it. To sum up, the action server do the following things:

- Create a list of all registered functioning units.
- In a loop routine, check the memory space(of data) of all plugs if they update the new data
- and transferring it according to the specified destination.

4.0.5 XML Synchronization

Easy to integrate is one of our main purposes which indicates we have quite flexibility dealing with modules. We allow different modules are able to exist on different terminals (e.g. vision module on computerA and navigation module on computerB). In order to fulfill the feature, we provide a synchronization method making the structure of every *QFlow* the same. The structure is saved in a XML file(Extensible Markup Language), every functioning unit's information will be saved, the name of a module, the number of plugs and Net Units IP address and port number etc.. See the example below:

Example of a XML file

```
<?xml version="1.0" ?>
<QFLOWDATA>
  <QFLOWWIDGET IP="127.0.0.1:10001" Path="./">
    <List Gr="11" />
      <cU Ty="1005" Name="Module_A">
        <BU Path="./Module_A.so">
      </cU>
      <cU Ty="1007" Name="result">
        <Conn ENDPOSX="407" ENDPOSY="205" .../>
      </cU>
      <cU Ty="1010" Name="192.168.1.10:10002">
         <NU IP="192.168.1.10" PORT="10002">
            <Plug Name="calculate_result" .../>
         </NU>
      </cU>
  </QFLOWWIDGET>
</QFLOWDATA>
```

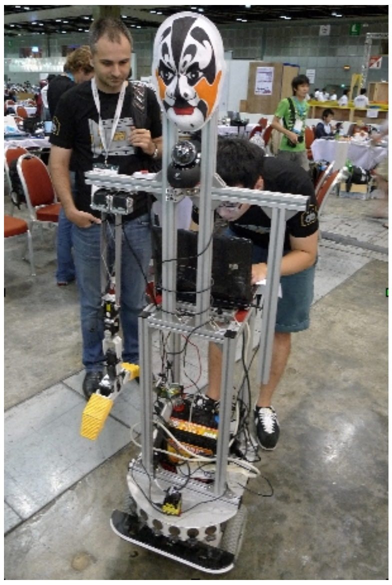
Figure 5: Our service robot.

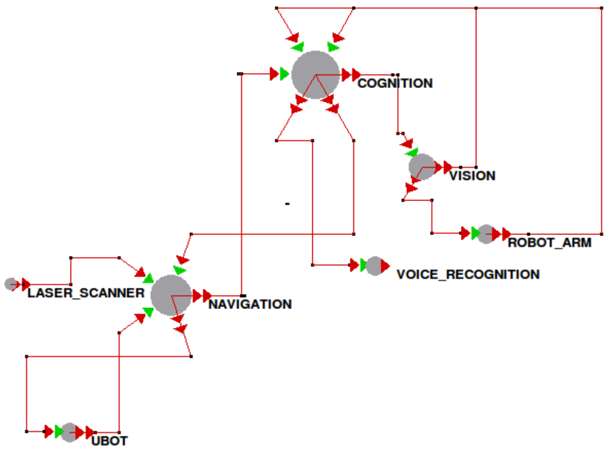

Figure 6: The *QFlow* structure of our robotic system.

(Label BU: Behavior Unit, Conn: Connector Unit, NU: Net Unit)

5 Result

We try to use *QFlow* with our service robot extensively (See Fig. 5). Fig. 6 is the basic structure of our robotic system. The cognition module is the center of autonomous decision, it sends the commands to other modules and wait for the task finished signal to arrive. Navigation module controls the movement of service robot(UBot, manufactured by ITRI[1]), in the mean while reading laser scanner data at all times for self-localization. Robot Arm module is controlled by vision module since the location of an object is determined by vision module, after robot arm either finished or unable to finish the task somehow, arm module will send a signal back anyhow, and wait for the next command.

6 Future Works

6.1 Reliability

Wireless communication is known to be less reliable and might cause significant bit-error rate. Hence, utilization of a wireless medium is becoming significant due to the limited bandwidth, especially in RoboCup competition, wireless communication is fundamental medium to many leagues[5]. Especially, UDP provides an unreliable service and we have not implement any checksum mechanism yet. We would like to do some reliability experiment on our *QFlow* server in the future to ensure the correctness of data sharing.

7 Conclusion

In this paper, we propose a program called *QFlow* which implement the data sharing method among the modules. *QFlow* helps the developers focus on their own module without worrying about data sharing problem. Furthermore, *QFlow* provides a good flexibility of structure, it synchronize throughout every terminal and allowing different modules can exist on different terminal, that is, making the integration of building up a system become easier.

Acknowledgement

The work has been supported by National Science Council grant umber 98-2218-E-194-003-MY2

References

[1] Industrial Technology Research Institute of Taiwan. *"http://www.itri.org.tw/"*, 2011

[2] Robocup@home, *"http://www.ai.rug.nl/robocupathome/"*, 2010.

[3] N. Michael Mayer and Li-Wei Lu and Yu-Min Hung and Hong Wu and Yu-Cheng Chang, Team *Crude-Scientists*, RoboCup Proceedings CD-Rom, 2010.

[4] N. Michael Mayer and Li-Wei Lu and Yu-Min Hung and Hong Wu and Yu-Cheng Chang, Using U-Bot for RoboCup@home, Proceedings of SICE Annual Meeting, Taipei, 2010.

[5] Frederico Santos and Luís Almeida and Luís Seabra Lopes and José Luís Azevedo and M. Bernardo Cunha, Communicating among Robots in the RoboCup Middle-Size League, pages 320-331, RoboCup, 2009,

[6] N. Yamasaki. Design and implementation of responsive processor for parallel/distributed control and its development environments. volume 13, pages 125-133, 2001.

[7] Yosuke Matsusaka, Kentaro Oku, and Tetsunori Kobayashi. Design and implementation of data sharing architecture for multifunctional robot development. volume 35, pages 54-65. Wiley Subscription Services, Inc., A Wiley Company, 2004.