A FRAMEWORK FOR SHARING INTELLIGENCE AMONG MOBILE ROBOTS ON A NETWORK

Naveen Kumar Toppo **Shivashankar B. Nair**
Dept of Computer Science and Engineering
Indian Institute of Technology Guwahati, Assam
INDIA, PIN – 781039
{toppo, sbnair}@iitg.ernet.in

ABSTRACT
The use of robots has dramatically expanded the potential of e-services. Intelligent robotic systems have been extensively used in a variety of different ways, both in industry and domestic life. In recent years, several researchers have been using the Internet as a command transmission medium to control intelligent robots and obtain feedback signals. This kind of system limits users to issue only dedicated commands and does not empower them to reprogram the system. This paper describes a framework to deploy robots, access them over a network and facilitate sharing of knowledge to realize an intelligent multi-robot society. The paper also discusses the development of a framework, which can aid in programming several robots. The framework can also coordinate their tasks by passing different degrees of intelligence using web services.

KEY WORDS
Robotics, Internet, Intelligence, Web Services.

1. Introduction

Recent advances in web technology have made it easier to realize a reliable and widely accessible communication framework. Remote control via the Internet is a comparatively young field of research, which has significant applications in the near future. Robotics, manufacturing, traffic control, space exploration, health care, disaster rescue, house cleaning, security, inspection and tele-presence are examples of such applications. Since the first robots appeared on the Internet in 1994 [1] an enormous amount of effort has been undertaken by several research laboratories to improve the underlying techniques. However, most of the initial robots on the Web have been of the industrial type operating within a very structured and contained environment [2]. Similarly, the first *mobile* robots on the Web have been mainly teleoperated systems [3] with very limited or no autonomy working in highly structured environments. With the advancement of technology, teleoperated systems and remote operating methods [4] have also changed their faces in the same pace. To get real unbound interaction with a distant environment, a mobile platform [5] is most adequate. It has no workspace limitations and thus allows for movement to places of interest and for real exploration of distant locations. A major breakthrough in this area has been the advent of technologies like Web Services in the .NET framework. Java and Jini have also contributed greatly to the domain of remote control. While past research has concentrated largely on robots with a centralized executive on board, current research has identified many benefits from *distributed* robotic systems [6],[7],[8]. Distributed systems are inherently more complex than single systems, introducing new challenges such as synchronizing the distributed set. All the robots currently deployed on the Internet are mainly teleoperated [9] systems, designed to to perform some predefined task. The user does not have much flexibility to maneuver the robot. These robots lack facilities for reprogramming and intelligence sharing. Data or knowledge acquired by them locally cannot be used by or shared with any other robots or devices on the Internet. The concept of embedding and sharing intelligence [10] among robots hosted on the Internet is still in its primitive stages. In this paper we endeavour this deficit by proposing a framework that takes into account the issues of storing and sharing information amongst robots deployed on a network.

The concept of web services in the .NET framework provides an ideal solution for accessing remote robots via a device independent platform. The web service applications also provide safe and secure connections with the robots. The framework does not confine itself to computers but also makes the whole operation inter compatible across devices.

The RobIN-II [Robots on the Internet] [11] is a framework for deploying robots and devices on a network. It is based on discernible functional decomposition and is suited against a stack of expertise levels. It enables users to command a variety of devices including robots and home appliances through normal desktop and smart devices. Users connect to a robot through a web interface and use the functionality supported by it.

The work described in this paper is an extension of this framework. While RobIN-II merely facilitates the deployment of devices, the enhancements described herein allow for sharing of knowledge and experience gained by

individual robots thereby culminating in the creation of a co-operative human-robot society over a network.

2. ROBIN II FRAMEWORK

RobIN-II [11] is primarily a peer-to-peer distributed architecture, with an additional specification of how to find one's peer. RobIN-II is based on the service oriented architecture (SOA) [12]. A wizard developed for RobIN-II, assists in the creation of template-files for the .NET environment with specifications that conform to the framework. RobIN-II follows a step-by-step improvement approach similar to the one used by Reid G. Simmons et al for Xavier [13], to make robots/devices appear on the Internet.

Using .NET as a base for development, the RobIN-II framework assumes three types of programmers-

1) The Base Level Programmer (BLP)
Base Level Programmers write device drivers for the specific machine or hardware. They constitute the programmers employed by the manufacturers of the robot/device who are fully aware of the hardware.

2) The Middle Level Programmer (MLP)
This programmer is proficient in at least one of the programming languages and is aware of the techniques used to couple the software (DLLs, APIs, etc.) supplied by the BLP, to finally create custom code for the final user. The wizard in Robin-II frees (to a great extent) the MLP from the arduous task of programming and deploying the device on the Internet.

3) The End User Programmer (EUP)
In RobIN-II, the EUP is basically a user commanding the robot through a client machine. A client could be a normal desktop machine or some other smart device connected to the Internet. It empowers the EUP to program the device for command and control, if certain provisions are made by the BLP and endorsed by the MLP.

3. *i*RobIN ARCHITECTURE

The core of the proposed framework christened ***i*RobIN** (Intelligent RobIN) lies in the underlying .NET technology which is the basis of the RobIN-II framework. By using RobIN-II as our base we have proposed an *intelligence* layer that can assist in sharing and passing information intelligently amongst the robots. Figure 1 depicts the overall architecture of ***i*RobIN**. All deployed robots strictly follow the proposed framework explained in the succeeding sections. Each host acts as a server and mediates an access with other deployed robots. Each robot has some basic and some complex functions depending on its purpose and capability.

Robots are deployed by the MLPs using web services that allow the EUPs to access and program them if required. Sharing of information gathered by each robot is achieved by a set of standardized web methods. These web methods can be consumed mutually by the robot programs. Their consumption by a third (human) party (EUP) can facilitate in multi robot programming over the network that can lead to a robot-human society on the network.

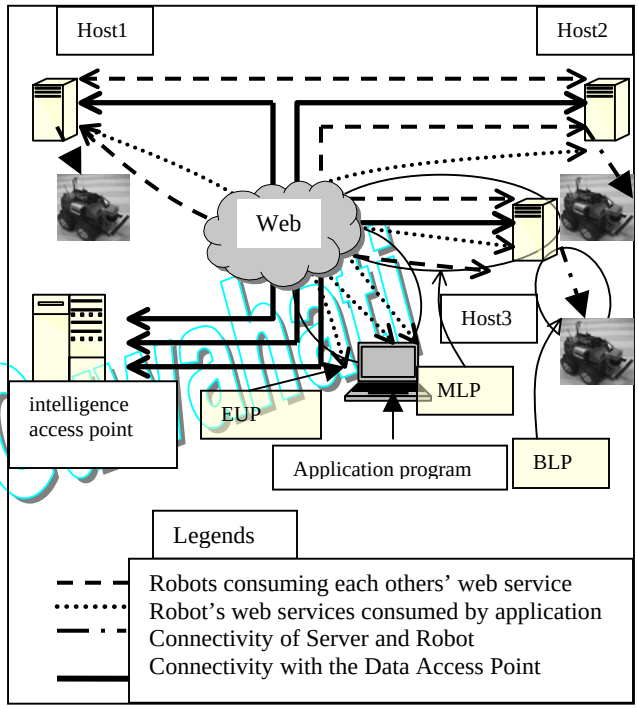

Figure 1 ***i*RobIN Architecture**

Each robot shares information on what it has learnt at the local machine that controls it. The word information could mean for instance the various obstacles and situations explored so far by a mobile robot. The MLP deploys the robots on the network through the server and publishes its capabilities and functions using web services, so as to enable its access remotely over the Web. The .NET Framework is facilitated by Web Service Description Language (WSDL). It describes the methods exposed by a web service and the number of parameters required by those methods. It provides the information using XML. Web services are advertised with the help of DISCO (Discovery of Web Services) so that a user can find these services on the Web. A user who wants to use the exposed web services consumes the DISCO file in his/her application to enable use of the given service. To conform to the ***i*RobIN** framework the MLP has to necessarily follow a set of specifications which help in transferring and interpreting information among the robots. The current version of the framework is being tested using several mobile robots (LEGO) each having a standard set of web services conforming to the framework.

4. KNOWLEDGE SHARING SCENARIO

Knowledge sharing is imperative to enable a society of robots to co-exist on a network. A typical scenario when robots can share information and co-exist is cited below.

A mobile robot R_1 stuck at some location is either forced to take a random move that could jeopardize its existence or ask for assistance from another robot the Web. Assuming it goes for the latter option, it is essential that R_1 search for such robots over the network that are similar to itself in structure and behave in the same or similar environment. Querying for help to robots whose structure and environments are vastly different, could result in another equally or more precarious situation. This calls for a network wide search for similar robots (structurally and environmentally). To achieve this every robot deployed in the ***i*RobIN** framework should host its configuration and environment information in such a manner that it is available to all other robots on the network. Robots also maintain a database that contains the information regarding the sensory conditions when they get trapped in a place and the action they took to recover from it. These databases are maintained on the local server hosting the robot and are updated on the fly. A search mechanism to find the best match yields the robot (R_2 for instance) that is most similar to the querying robot R_1. The database in the server hosting R_2, is searched for the set of conditions in which R_1 presently is. If a match is found the task which R_2 performed for these conditions is mimicked by R_1 thereby sharing the experience of R_2.

4.1 Specifications

***i*RobIN** prescribes that any robot embedded on a server should publish its web service and maintain all the information in its local database. The meaning of the word *information* has to be interpreted based on the perception of robot, i.e., in the way the robot views and interacts with the real world. It pertains directly to the sensors on board the robot and the tasks it executes. For instance during the task of exploration if a robot detects an obstacle in the front, it will make an entry of this information in its database specifying the kind of obstacle (static/dynamic) encountered and the side where it was detected. More detailed and relevant information can also be stored, depending on the capabilities of the robot.

4.2 Mandatory Functions

The ***i*RobIN** framework enforces all MLPs to expose some mandatory functions and also perform book keeping as well as information of the configuration of the robot being deployed and its environment, details of which are exemplified below-

4.2.1 Configuration Information

This information includes the name of the robot, its type and number of actuators and sensors. It also includes the information about the details of the orientation of the sensor. The information is stored in a file using XML. Contents of a sample robot configuration is depicted in Figure2.

```
<!DOCTYPE config>
<config>
    <MAKE>LEGO</MAKE>
    <ROBOTID>Mylego</ROBOTID>
   <STRUCT>ROVER</STRUCT>
    <SENSORS>3</SENSORS>
    <TOUCH1>LEFT</TOUCH1>
    <TOUCH2>RIGHT</TOUCH2>
    <LIGHT1>DOWN</LIGHT1>
    <MOTOR1>OUTA</MOTOR1>
    <MOTOR2>OUTC</MOTOR2>
</config>
```

Figure 2 **A Sample Robot Configuration**

The meanings of each of the mandatory tags used are given below –

MAKE :- Specifies the manufacturer's name.
ROBOTID :- This is used to identify the robot
STRUCT : - It describes the structure of the robot (like rover, caterpillar etc.)
SENSORS :- quantity of sensors
TOUCH1 :- Touch sensor
TOUCH2 :- Touch sensor
LIGHT1 :- Light sensor
MOTOR1 :- Motor1
MOTOR2 :- Motor2.

4.2.2 Environment Information

This conveys the nature about the environment of the robot. The environment information, is also stored as an XML file a sample of which is depicted in Figure 3

```
<environment>
    <shape>RECTANGLE</shape>
    <area>2500sqcm</area>
   <objects>
      <static>3/NULL</static>
      <dynamic>NULL</dynamic>
    </objects>
 </environment>
```

Figure 3 **A Sample Representation of a Robot Environment**

The environment information includes among other aspects, details like the shape, number of obstacles detected so far and their nature (dynamic or static).

4.2.3 Web services at the Robot Hosts

At each robot host some mandatory web services are hosted to facilitate the access of the database, configuration and environment information. The details of these functions hosted as web services are cited below.

a) Information retrieval function (*share()*)

This function retrieves the solution from the database at the robot server hosting this service and passes it on to the requester. For the Lego robot society this function takes sensor values as input parameters and returns the corresponding action taken by the local robot.

b) Robot configuration retrieval function (*config ()*)

The function is used to procure the configuration information of the robot. This function takes no parameters.

c) Robot environment retrieval function (*enviro ()*)

This function provides access to the environment information of the local robot and does not take any parameters.

d) Database at the Robot host

This database is an XML file that contains the information on what the robot has learnt so far. One such file for a LEGO robot is shown in Figure 4. The contents can be interpreted as the rule - *if TOUCH1 = 1 and TOUCH2 = 0 and LIGHT = 55 then action is TURN-RIGHT*. The contents are constantly updated as the robot learns to survive in its environment using the MLP deployed custom learning algorithm.

```
<info>
<TOUCH1> 1 </TOUCH1>
<TOUCH2> 0  </TOUCH2>
<LIGHT1>  55 </LIGHT1>
 <action>  RIGHT  </action>
</info>
   .
   .
```

Figure 4 **A Sample Database file for LEGO robot**

5. INTELLIGENCE ACCESS POINT

The Intelligence Access Point (IAP) forms the hub of the *i***RobIN** framework. The IAP is hosted on a server and facilitates searching for the solutions to queries from the robots in the network. Figure 5 depicts the working of the IAP. The dashed lines show the flow of information.

Every robot in the network consumes the dedicated web services hosted at the IAP. Given the configuration, environment information and sensor conditions of the querying robot, the web services at the IAP searches the network for the best matched robot and returns the solution along with the corresponding name of the robot server that supplied the information.

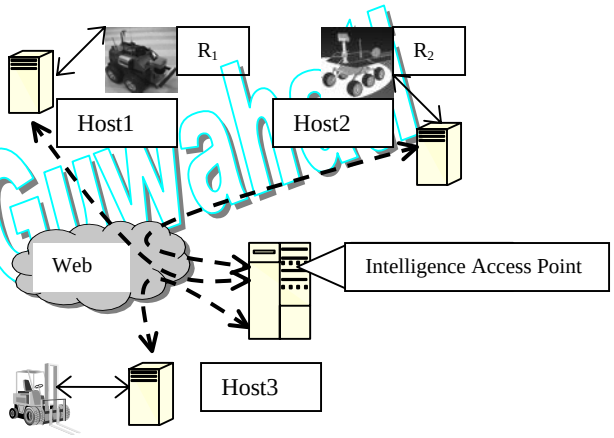

Figure 5 *i***RobIN - Sharing Intelligence**

The search is performed by initially, procuring the configuration and environment information of all the connected robots to the network and then finding the one that is most similar to the querying robot. This assures that there is a good degree of resemblance between the querying robot and the robot that is offering a possible solution, both in the terms of configuration and environment. The best match amongst configuration is found using equation 1.

$$W_C = \sum_{i=1}^{N} S_i \qquad (1)$$

Where $S_i = 1$ if the i^{th} sensor of both the robots are mounted in the same fashion, else $S_i = 0$. N is the number of sensors on the querying robot. The final "distance" between the environments of two robots is found using the equation 2

$$W_E = \{ A \times (x \times \sum_{i=0}^{n} S_i + y \times \sum_{i=0}^{n} D_i) \div (\sum_{i=0}^{n} S_i + \sum_{i=0}^{n} D_i) \} + G$$

(2)

where *A* is the area of the environment, *S* and *D* are the number of static and dynamic objects, discovered by the robot to date, respectively and *x* and *y* are constants that weight the number of these objects. We have taken the values of *x* and *y* as 1 and 2 respectively in our current studies. The value of G depends on whether the two environments match. G equals 1 if it is an exact match, else it is taken to be 0. The total distance is calculated based on the sum of the distances between the configuration and environment information obtained using equations 1 and 2.

Total distance = $W_C + W_E$

Once the best match is found the IAP searches the local database of the concerned robot for sensor conditions identical to that of the querying robot. It retrieves the action taken for this combination and returns it to the querying robot as the solution.

The IAP maintains a list of the best matches in order to enable the querying robot to ask for information from the remaining set of robots in case the solution from the first one fails.

The IAP hosts the following web services for realizing the above searching and sharing of information. The services can also aid the deployer to write his own custom search mechanism. (C and E stand for configuration and environment information respectively, S the sensor values and R the name of the server which serves the solution.)

1. get_solution() - The querying robot uses this service to find a solution to its problem. The service accepts the E, C and S values from the robot, searches and sends back the action to be taken along with R.

2. get_all_information() - This web service takes no input parameters and returns the C and E of all robots on the network together with their respective R values.

3. get_solution_from_server () - It takes as input a specific R value and the current S of the querying robot and retrieves a solution from R.

6. WORKING OF FRAMEWORK

The two scenarios described in Figure 6 and Figure 7 show a broad view of the working of the framework. Scenario 1 depicted in Figure 6 throws more light on the working of the ***i*RobIN** framework. Three robots connected to the three different servers form the robot hosts. WS_i denotes the web services at host *i*. In a typical case, the robot at host1 could be one with a wandering behaviour while that at host2 could be exhibiting photo taxis. If the wandering robot gets stuck at some place in its environment, it requests for help via the IAP. The latter in turn finds the best-matched robot and provides a solution, if it exists from within this robot's database. If a valid solution was provided from the database of the robot at host2, the robot1 builds its trust level on this robot else it reduces it below a threshold value.

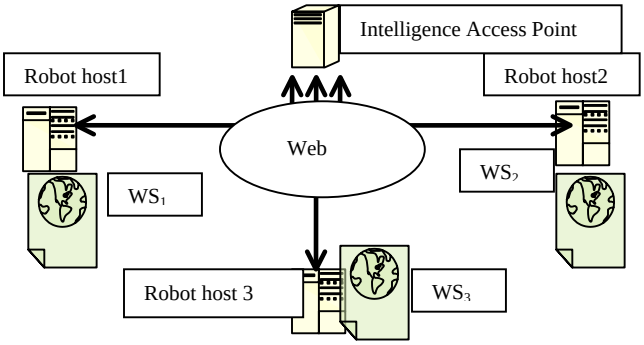

Figure 6 ***i*RobIN - Scenario 1**

If the robot at host1 again requires a solution it consults the trust levels it has built and requests the IAP to provide a solution from this specific robot. If the wandering robot at host1 were to continuously consult the one at host2, it may as well forget wandering and learn to exhibit photo taxis as and when it get stuck, thereby generating specific skills.

Scenario 2 in Figure 7 brings out the scalability of the framework. If the deployer of robot at host3 wishes, he may consume the web services of the robots at host1 and 2 to realize his own search strategy thereby bypassing the mechanisms hosted at IAP.

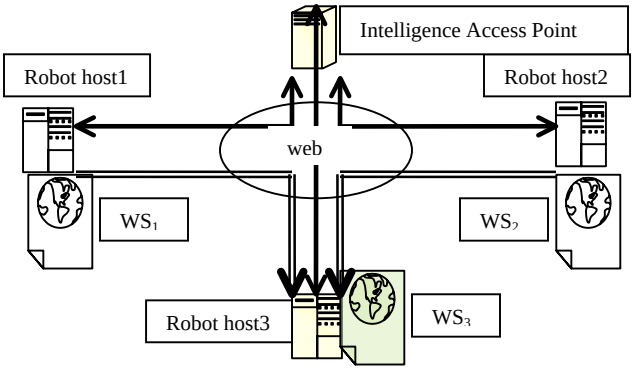

Figure 7 ***i*RobIN - Scenario 2**

7. Conclusions

While the RobIN-II framework concentrated on deploying robots on a network, its extension ***i*RobIN** focuses on intelligence sharing over the network. The framework is scalable in the sense that deployers can add new features and host them as web services to allow other deployers to consume them. The IAP serves as a starting mechanism for ***i*RobIN** providing the bare initial web services. On our test bed we have currently deployed three Lego robots, each embedded with a unique behaviour. Investigations on how intelligence, acquired using Artificial Immune Systems

[14], can be augmented to the ***i*RobIN** framework, is in progress.

References:

[1] K. Goldberg, Desktop Teleoperation via the World Wide Web. *Proc. IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995, 654-659.

[2] Ken Goldberg, Billy Chen, Rory Solomon, Steve Bui, Bobak Farzin, Jacob Heitler, Derek Poon, Gordon Smith, Collaborative Teleoperation via the Internet. *Proc. IEEE International Conference on Robotics & Automation*, San Francisco, CA, 2000, 2019-2024.

[3] O. Michel, P. Saucy, and F. Mondada, Khep-OnTheWeb: An Experimental Demonstrator in Telerobotics and Virtual Reality. *Proc. Virtual Reality and Multimedia Conference, IEEE Computer Society Press*, Switzerland, 1997, 99-115.

[4] Dirk Schulzy Wolfram Burgardz Armin B. Cremers, Robust Visualization of Navigation Experiments with Mobile Robots over the Internet. *Proc. IEEE/RSJ International Conference on Intelligent Robotics and Systems*, Kyongju, Korea, 1999, 942-947.

[5] Sukhatme, G.S. and Mataric, M.J, Embedding Robots Into the Internet. *Communication of the ACM*, 43(5) 2000, 67-73.

[6] Gregory Dudek and Michael Jenkin, A Multi-Layer Distributed Development Environment for Mobile Robotics. *Proc. International Conference on Intelligent Autonomous Systems,* , Pittsburgh, PA, 1993, 542-550.

[7] http://ic.arc.nasa.gov/projects/psa/

[8] Pui Wo Tsui and Huosheng Hu, A Framework for Multi-robot Foraging over the Internet. *Proc. IEEE International Conference on Industrial Technology*, Bangkok, Thailand, 2002, 897-902.

[9] Tse Min Chen, Ren C Luo, Remote Supervisory Control of An Autonomous Mobile Robot. *Proc. IEEE Internat. Symp. on Industrial Electronics, ISIE'97-* Guimaraes, Portugal, 60-64.

[10] Liam Cragg, Pui Wo Tsui and Huosheng Hu, Building a Fault Tolerant Architecture for Internet Robots using Mobile Agents. *Proc. 1st British Workshop on (IORW)*, Reading, May 2003.

[11] Rajendra P. Badapanda, Shivashankar B. Nair, Dong Hwa Kim, A Framework for Rapid Deployment of Devices and Robots on a Network. *Proc. 1st International Computer Engineering Conference*, 2004, Cairo, Egypt, 625-629.

[12] M.P. Papazoglou and D. Georgakopoulos, Service Oriented Computing. *Communications of the ACM*, 46(10), 2003, 25-28.

[13] Simmons Reid, Fernandez J., Goodwin R., Koenig S. and Sullivan J.O, Xavier, An Autonomous Mobile Robot on the Web. *Robotics and Automation Magazine*, 1999. Available: http://www.cs.cmu.edu/~Xavier

[14] A. Ishiguro, Y. Watanabe, T. Kondo and Y. Uchikawa, Immunoid, A robot with a Decentralized Behavior Arbitration Mechanism Based on Immune System. *Proc. 4th International Conference n Automation, Robotics and Vision*,1996, 1600-1605.