International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

A Service Oriented Middle-Ware for Managing Execution Sequence of Distributed Robots through Orchestration

Shweta Agrawal[Å*] and Rajkamal[Å]

[Å]School of Computer Science & Information Technology, DAVV, Takshashila Campus, Khandwa Road, Indore (M.P.) 452017 India

Accepted 10 August 2014, Available online 25 Aug2014, **Vol.4, No.4 (Aug 2014)**

Abstract

An Orchestrator coordinates and control computations in parallel and sequential computing nodes. The paper presents an Orchestrator framework for automation of multiple robots. The orchestrator controlling mechanism is adaptive and flexible. According to application, it is able to decide the invoking sequence of robots. The proposed framework also manages service failures. The framework is implemented using RMI, multithreading and RS-232 standard in Java.

Keywords: *Orchestrator, multiple robots, distributed computing, multithreading, RMI.*

1. Introduction

Numerous applications need use of autonomous robots. Applications such as cleaning of toxic waste, nuclear power plant decommissioning, search and rescue missions, surveillance, and survey tasks requires a lot of manpower, time and have elements of danger. A coordinator manages the activities of distributed robots. The coordination and management of robots is very difficult. The use of a set of distributed robot systems reduces cost, time and provides reuse of robots. A single robot can also consist of multiple robotic components. The Player [Nakamura A. et al, 2002], ROCI [Murphy R. et al, 2002], are the some frameworks which are using bottom up programming approach.

There are a lot of issues in distributed multiple robots and multiple component systems. These are synchronization, communication, planning and architectural issue, centralized, and decentralized coordination. The technique of integrating the various robots/components enable distributed execution essentially transparent to the component developer.

The paper proposes an orchestrator framework for managing multiple robots/components. Orchestrator [Peltz, C, 2003] [LavnyaRamakrishnan et al, 2011] integrates, manages and coordinates the functioning of multiple robots/components. Orchestrator can decide the calling sequences of multiple robots /components according to requirement of an application. The Orchestrator manages timeouts, priority and service failures also. Proposed framework is flexible and programs the calling sequences of robots/components according to the requirements.

The paper organized as follows: Section 2 describes related work in the area of multiple robots; Section 3 is presenting the proposed framework; Section 4 gives the implementation procedure; Section 5 gives the conclusion from the study.

2. Related Work

Developing architectures for distributed robotics includes a great deal of research. The main focus of these architectures are on planning and controlling the task of distributed robots. The main issues are selecting actions, the way of communication, controlling, heterogeneity and homogeneity of robots, synchronization, solving conflicts and many more. The focus of a developed architecture for multi-robot teams tends to provide a specific type of potential to a distributed robot team. The objective of the authors [R. Alami et al, 1997] is task planning. It showed architecture for the control of a large fleet of autonomous mobile robots. It found applications for containers, trans-shipment, airports and marshalling yards. The authors [L. E. Parker, 1998] described the control architecture ALLIANCE. It has feature of fault tolerance, reliability, and adaptive cooperation among small to medium size heterogeneous mobile robots. The authors [Cowley et al,2007] used a concept of orchestration which controls the swarm robots. The paper specified a reactive behavioral configuration for a multi agent team. The team requires a careful choice of the behavior set and the creation of a temporal chain of behaviors in order to finish a mission. This difficult task is simplified by applying an object-oriented approach to the design of the mission and by using a methodology called temporal sequencing. The authors [D. MacKenzie et al, 1997] described the mission plans based on object oriented approach. The authors [P. Stone et al,1999, C. Candea,2001 and E. Pagello et al,2002] described the assignment of the role of the team. The authors [Rybski et al, 2002] presented software architecture for the control of a set of small robots, called

*Corresponding author: **ShwetaAgrawal**

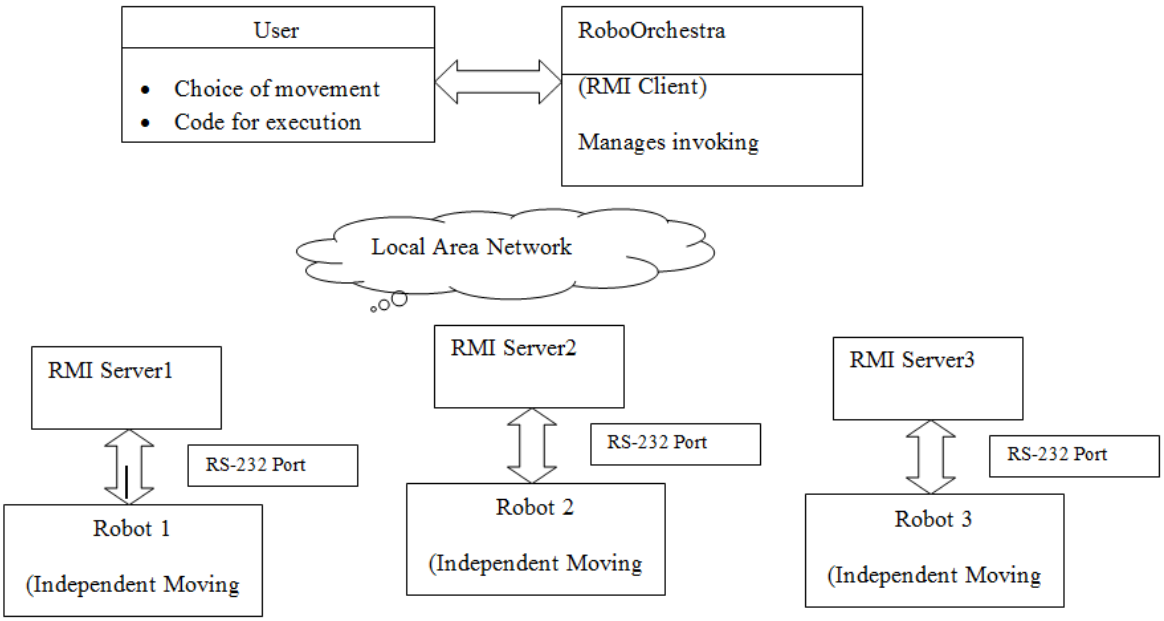

Figure 1: RoboOrchestra application Environment

Table 1: Movement of Robots

S. No.	Code	Action	Shape of path
1	FLBRS	Forward-Left-Backward-Right-Stop	Square
2	FBS	Forward-Backward-Stop	Stripped vertical
3	BFS	Backward- Forward-Stop	Stripped vertical
4	LFS	Left -Forward –Stop	L shape unidirectional
5	RFS	Right-Forward-Stop	L shape unidirectional
6	LRS	Left Right-Stop	Stripped horizontal
7	RLS	Right-Left-Stop	Stripped horizontal
8	BS	Backward-Stop	Stripped vertical unidirectional
9	FS	Forward-Stop	Stripped vertical unidirectional
10	RS	Right –Stop	Stripped horizontal unidirectional
11	LS	Left –Stop	Stripped horizontal unidirectional

Table 2: Code of Execution

S. No.	Nature of code	Example	Action	Mode of execution
1	Incremented	1 2 3 or 4 7 9 or 3 8 9 or 2 5 8	Invoke Robot 1first after time completion of robot 1 invoke robot 2,after time completion of robot 2 invoke robot 2	Sequential execution
2	Decremented	7 3 2 or 6 3 1 or 75 63 2		Sequential execution
3	Same value	1 1 1 or 2 2 2 or 8 8 8 or 9 9 9	Invoke all robots simultaneously	Parallel execution
4	Some values are same	1 1 3	Invoke robot 1 and 2 in parallel and invoke robot 3 after time completion of robot 1 and 2	Hybrid execution
5	Some values are same	3 2 2	Invoke robot 1 first, then invoke robot 2 and 3 in parallel after time completion of robot 1	Hybrid execution

Scouts. The architecture for this system constrains by the limited computational capabilities of the small robots. It used a proxy-processing scheme. Robots use remote computers for their computing needs. The architecture design challenge is also addressed in [LavnyaRamakrishnan et al, 2011]. It presented a flexible command and monitoring structure that enables a human operator to work with a team of mobile robots. The paper [Fumio et al,2008] proposed an action framework for resolving the given command into sequence, which will be performed by robots. A control layer for providing adaptive behavior of process and motion planning in industrial robots has been introduced in paper [Lambrecht, J et al, 2011].

3 .The Proposed Framework

The framework is based on networked architecture. There is *n* number of machines; each machine in network has a robot connected via RS-232 standard. The robots in

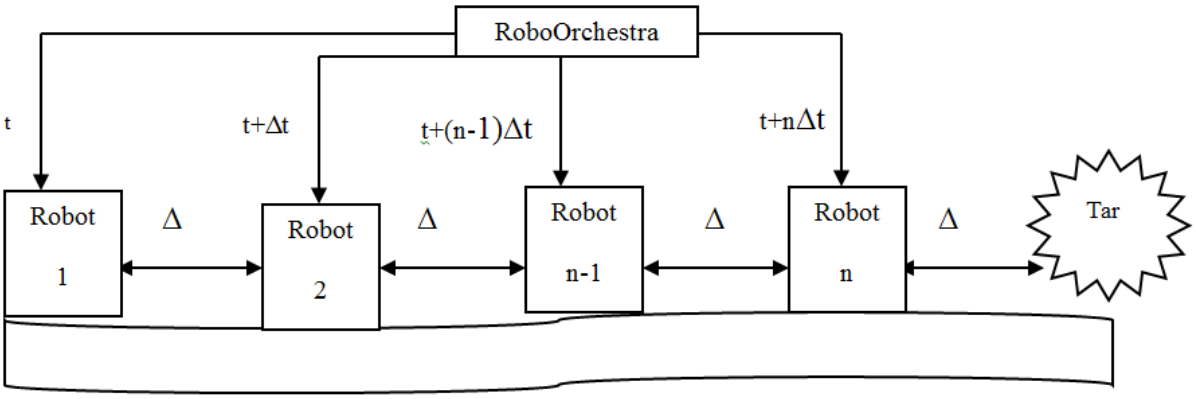

Figure 2: Sequential execution of robots

network can be controlled by a single machine. The controlling machine has the software named as RoboOrchestra. It is orchestrating the activities of multiple robots/components to achieve a common goal. All machines are linked together through Ethernet LAN.

The RoboOrchestra sends controlling signal to the networked computer and the computer will forward it to serial port to govern the attached robot. The actual control will be done by RoboOrchestra. The framework is presenting the flexible and fault tolerance mechanism for controlling the multi robots/ components. The framework is providing the facility of a sequential, parallel or hybrid invocation of robots/components. The RoboOrchestra can invoke robots in sequential, in parallel or in hybrid manner. For managing fault tolerance, if one of the robots fails to respond, it can invoke its alternative robot for the same task. Fig.1 shows the application environment for the RoboOrchestra. It is acting as a central coordinator, which is interacting with user and remote robots. The user can enter choice of movement and a code for deciding the sequence of robots. The RoboOrchestra is managing the sequence of execution, and exceptions. Robots are the moving vehicles. They can move forward, backward, left and right.

Table 1 gives the different movements of robots and their corresponding code. Table 1 also gives the shape of path for each code. User may choose different codes for different shapes of path. Table 2 gives the codes for choice of execution of multiple robots. Continuous incremented or decremented integers will lead to the sequential execution of robots. The same value of all integers will lead to parallel execution of robots. For the code which is not exactly incremented or decremented, RoboOrchestra will perform hybrid execution. Example includes some integers have same value and other have different values, then for the same value of integers, RoboOrchestra will invoke robots in parallel and for other values, it will do sequential execution.

3.1 Sequential Execution

Sequential execution is based on two factors: Time based execution, and response based execution. There would be a fixed time span for each robot in time based execution. A number will be assigned to every robot. A robot that has assigned number 1 will be invoked first, after completion of time span of robot 1, robot 2 will be invoked, after completion of time span of robot 2, robot 3 will be invoked and continues till last one.

A sequence number will be assigned to every robot, in response based execution also. The robot 1 will be invoked first, after getting a response from robot1, robot 2 will be invoked, after getting a response from robot 2, robot 3 will be invoked and so on. It is possible in response based execution that one of the robots is not responding and hanging the overall system. For handling this situation, the RoboOrchestra will wait for certain amount of time for response and after that will invoke its alternative robot for the same task.

Considering a case of rescue operation, where the path is very narrow, at a time only one robot can travel, every robot can travel only limited distance and the initial location of all robots are fixed. Here aim is to send an object to the target. Fig.2 shows the arrangement of multiple robots in a sequential manner to reach the target. The RoboOrchestra is aware about the speed of the robots. Based on the speed and distance travelled by the robots, RoboOrchestra calculates the time span taken by each robot. Here we assume that the speed of all robots is same and robots will take Δt time to travel the distance Δd. RoboOrchestra will invoke robot 1 at time t, robot 1 will travel the distance Δd in time Δt and will transfer its object to robot 2, then RoboOrchestra will invoke robot2 at time $t+\Delta t$, robot2 will travel distance Δd in time Δt and will transfer its object to robot 3 , at time $t+2\Delta t$ RoboOrchestra will invoke robot3. This sequence would be continued for all the robots. At time $t+(n-1)\Delta t$ RoboOrchestra will invoke robot n. The last robot will reach to target at time $t+n\Delta t$. After transferring object to the next robot all robots will take their initial positions, so that they will be available for their next application. The movement would be in both forward and backward directions.

The long distance may be spanned by multiple robots by using orchestration; even the individual robot has limited traveling capacity. If the traveling capacity of each robot is different then also the RoboOrchestra can assign different time spans to the robots to achieve the common goal.

3.2 Parallel Execution

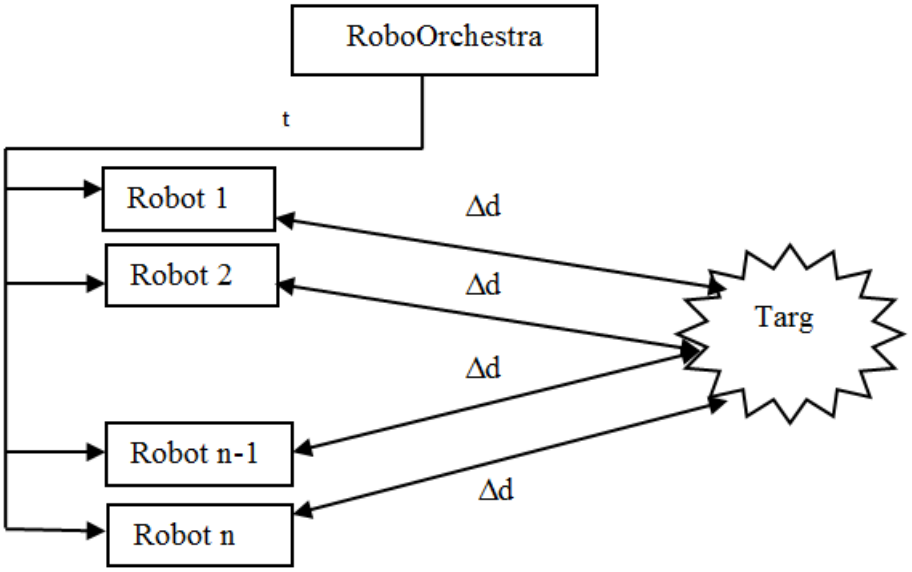

Figure 3: Parallel execution of robots

Table 3: Sequence of Execution By RoboOrchestra

Robot/ Time instance	1	2	3	4	5	6	7	8	9	10	11	12	Invoke
T	O-1	O-2	O-3	O-4	O-5(H)								F
													B
t+Δt						O-1	O-2	O-3	O-4(H)				F
	√	√	√	√									B
t+2Δt	O-5	O-6	O-7							O-1	O-2	O-3	F
						√	√	√					B
t+3Δt (O-1,O-2 and O-3 have been reached to the target						O-4	O-5	O-6	O-7(H)				F
	√	√	√							√	√	√	B
t+4Δt	O-8	O-9	O-10							O-4	O-5	O-6	F
						√	√	√					B
t+5Δt (O-4,O-5 and O-6 have been reached to the target						O-7	O-8	O-9	O-10 (H)				F
	√	√	√							√	√	√	B
t+6Δt	O-11	O-12	O-13							O-7	O-8	O-9	F
						√	√	√					B
t+7Δt (O-7,O-8 and O-9 have been reached to the target													F
	√	√	√			O-10	O-11	O-12	O-13 (H)				B
t+8Δt	O-14	O-15								O-10	O-11	O-12	F
						√	√	√					B
t+9Δt (O-10,O-11 and O-12 have been reached to the target						O-13	O-14	O-15					F
	√	√								√	√	√	B
t+10Δt										O-13	O-14	O-15	F
						√	√	√					B
t+11Δt													F
										√	√	√	B
t+12Δt	Robot 10 , robot 11 and robot 12 will take their initial position												

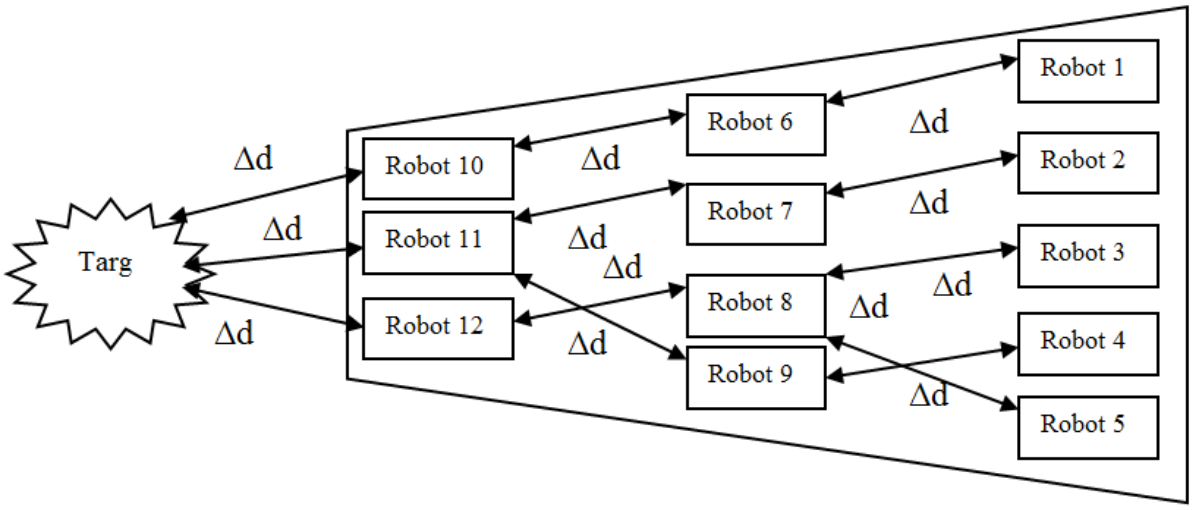

Figure 4: Hybrid execution of robots

Table 4: Sequence of Execution without Orchestration

Time Instance/ Robot	1	2	3	4	5	6	7	8	9	10	11	12	Invoke
t	O-1	O-2	O-3										Forward
													Backward
t+Δt						O-1	O-2	O-3					Forward
													Backward
t+2Δt										O-1	O-2	O-3	Forward
													Backward
t+3Δt													Forward
										√	√	√	Backward
t+4Δt													Forward
						√	√	√					Backward
t+5Δt													Forward
	√	√	√										Backward
t+6Δt	O-4	O-5	O-6										Forward
													Backward
t+7Δt						O-4	O-5	O-6					Forward
													Backward
t+8Δt					O-4					O-4	O-5	O-6	Forward
													Backward
t+9Δt													Forward
										√	√	√	Backward
t+10Δt													Forward
						√	√	√					Backward
t+11Δt													Forward
	√	√	√										Backward
t+12Δt	Robot 1, robot 2 and robot 3 will take their initial position												

Multiple robots can be invoked concurrently in parallel execution. Fig. 3 shows the parallel execution of multiple robots, which is controlled by RoboOrchestra. Four robots have been shown in Fig. 3. They all have to travel distance Δd to reach to the target. RoboOrchestra will invoke all four robots at time t and will get response from all robots at time t+Δt.

Considering the case of rescue operation, where the path is wide enough. RoboOrchestra can invoke multiple robots in parallel to send the food packets or other relief material to the people. With parallel execution of robots speed of work can be increased.

3.3 Hybrid Execution

Execution in hybrid mode consist the combination of serial and parallel execution both. Again considering the case of rescue operation where, the objective is to send n objects to the target, initially; the path is wide, but it is becoming narrow slowly. There are k robots to execute this task, every robot can travel only Δd distance, the path is becoming narrower at distance Δd. It is becoming again narrower at distance 2Δd. Assuming that path is becoming narrower till three levels. The width of path is reducing in a manner that after each level one robot would be strike

out from the path. Fig. 4 shows the arrangement of 12 robots on the assumed path. The formula for sending n objects through k robots and settling down to all the robots to their initial position is:

$$t + \left(\frac{2n}{\frac{k}{3}-1} + 2\right)\ \Delta t \qquad (1)$$

n should be in multiple of 3. If objects are not in multiple of three, then calculation will be done by assuming nearest higher value of multiple of three. For example if objects are 14, calculation will be done for 15 objects.

The Table 3 gives the sequence of execution of robots to reach to the target. A sequel shows all parallel operations that can be invoked in that time instance. It has been assumed that for traveling Δd distance all robots will take Δt time. t is taken as the starting time instance. Table 3 shows the sequels for sending 15 objects through 12 robots.

Abbreviations and symbol used in Table 3 are as follows:

O: Object
√: The robot will be invoked in specified direction
H: The object is on hold.
F: Invoke Forward
B: Invoke Backward

Description of some sequels from Table 3 is as follows: at time t robot 1, robot 2, robot 3 and robot 4 will be invoked in forward direction to transfer the object 1.object 2, object 3 and object4 respectively to the next robots. Robot 5 will hold the fifth object. In sequel 2 at time t+Δt robot 1, robot 2, and robot 3and robot 4 will transfer their objects to robot 6, robot 7, robot 8 and robot 9, and will be invoked in backward direction. In sequel 3, at time t+2Δt robot 1, robot 2 and robot 3 will be invoked in forward direction to transfer the object 5, object 6 and object 7 to the next robots; robot 6, robot 7 and robot 8 will be invoked in back direction and robot 10, robot 11 and robot 12 will be invoked in forward direction to transfer the object 1, object 2 and object 3 to the target. In sequel 4 at time t+3Δt, robot 1, robot 2 and robot 3 will be invoked in backward direction; robot 6, robot 7 and robot 8 will be invoked in forward direction to transfer the object 4, object 5and object 6 to the next robots; robot 9 will hold the object 7; robot 10, robot 11 and robot 12 will be invoked in backward direction. Object 1, object 2 and object 3 will be on the target.

Table 4 gives the sequels without orchestration**.** The formula for sending n objects through k robots and settling down the robots to their initial position is:

$$t + \left(\frac{6n}{\frac{k}{3}-1}\right)\ \Delta t \qquad (2)$$

The time relation 1 and 2 shows that RoboOrchestra is performing much faster than un Orchestrated approach.

4. Implementation

The framework RoboOrchestra has been implemented using Remote Method Invocation (RMI) and serial port communication in Java. RMI has been used to invoke the services from remote machines. Multiple threads execute the services in parallel or sequential. Four servers RoboServer1, RoboServer2, RoboServer3 and RoboServer4 have been developed. RMIClient is playing the role of RoboOrchestra. It has a stub object of the servers and servers have the skeleton object of RMIClient. The stub object consists of an identifier of the remote object to be used, an operation number describing the method to be called and the marshalled parameters. RMIClient sends all these information to the server. The job of skeleton object at the server side are unmarshalling the parameters, calling the desired method on the real object lying on the server, capturing the return value or exception of the call on the server, again marshalling the return value, sending a package consisting of the value in the marshalled form back to the stub on the RMIClient. At RMIClient thread has been developed to call four servers in parallel or in sequential manner. For managing the sequence of operations synchronized methods have been used. RoboOrchestra send the input data to all four servers and the servers are transferring these control information to the robots, which are connected through the serial port of computer.

According to user input the RoboOrchestra decides the sequences of signals which would be transferred to the robots.The experimental setup was tested with three machines, one client, one local host and two servers were made for implementing the framework. The configuration of machine was Intel (R) core (TM) 2 Duo CPU with 2 GB RAM and 2.93 GHz clock frequency. The Figure 5 and 6 are showing the experimental robots and their connection with computer.

Figure 5: Experimental Setup

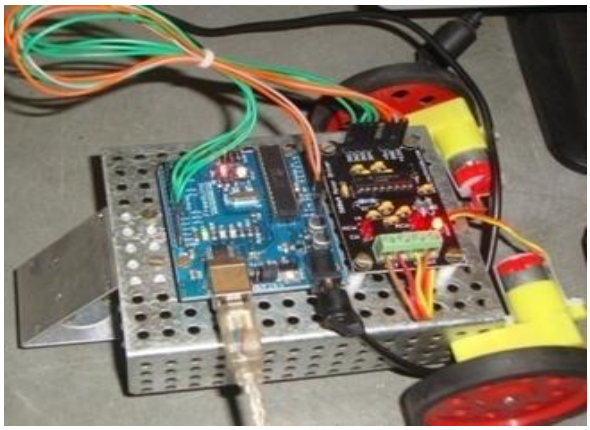

Figure 6: Robot used for experiment

Conclusion

A framework RoboOrchestra for controlling distributed robots has been presented. It is a service oriented middleware, which is controlling the execution sequences of robots according to the application. The proposed framework also provides flexibility of movements of robots and execution sequence. It has been proved that RoboOrchestra performs very fastr in hybrid execution of robots. It is managing service failures also, by invoking an alternative service for the same task. The framework has been implemented using remote method invocation, serial port communication and multithreading in java.

References

Nakamura, A., Ota, J., Arai, T. (Oct 2002), Human-supervised multiple mobile robot system, Robotics and Automation, IEEE Transactions on , 18(5), pp.728,743.

Murphy, R.R., Lisetti, C.L, Tardif, R.,Irish, L, Gage, A., (Oct 2002),Emotion-based control of cooperating heterogeneous mobile robots, Robotics and Automation, IEEE Transactions on ,18(5), pp.744,757.

Peltz, C., (Oct. 2003), Web services orchestration and choreography, Computer, 36(10), pp.46- 52

LavnyaRamakrishnan, Jeffrey S. Chase, Dennis Gannon, Daniel Nurmi, Rich Wolski "Deadline-sensitive workflow orchestration without explicit resource control" Elsevier J. Parallel and Distributed Computing Vol. 71, No. 3, pp. 343-353, March 201

R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. (1997), Multi-robot cooperation in he Martha project.IEEE Robotics and Automation Magazine

L. E. Parker. (1998) ALLIANCE: An architecture for fault-tolerant multi-robot cooperation. IEEE Transactions on Robotics and Automation, 14, pp. 220–240.

Cowley, Anthony, Taylor, C.J.,(2007), Orchestrating concurrency in robot swarms, IEEE/RSJ International Conference onIntelligent Robots and Systems, pp.945,950,.

D. MacKenzie, R. Arkin, and J. Cameron.(1997) Multiagent mission specification and execution. Springer J. on Autonomous Robots, 4(1), pp. 29–52,

P. Stone and M. Veloso.(June 1999), Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Elsevier J. Artificial Intelligence, 110(2), pp.241–273.

C. Candea, H. Hu, L. Iocchi, D. Nardi, and M. Piaggio,(August 2001), Coordinating in multi-agent robocup teams. Robotics and Autonomous Systems, 36(2- 3), pp.67–86.

E. Pagello, A. D'Angelo, C. Ferrari, R. Polesel, R. Rosati, and A. Speranzon, (2003.), Emergent behaviors of a robot team performing cooperative task, Taylor and Francis J. On Advanced Robotics ,17(1),pp.3-19.

Rybski, Stoeter, Gini, Hougen, and apanikolopoulos ,(Oct 2002), Performance of a Distributed Robotic Systems Using Shared Communications Channels , IEEE Trans. on Robotics and Automation ,18 (5) ,pp. 713-727.

Fumio, O. Toshiba Corp., Kawasaki Junichiro, O., (Dec. 2008) ,An Action Framework for Robots based on Distributed Knowledge Base, IEEE/SICE International Symposium on System Integration, pp. 77 – 82.

Lambrecht, J., Dept. of Ind. Autom. Technol., Tech. Univ. Berlin, Berlin, Germany , Chemnitz, M. , Kruger, J.(April 2011), Control layer for multi-vendor industrial robot interaction providing integration of supervisory process control and multifunctional control units,IEEE Conference on Technologies for Practical Robot Applications (TePRA), pp. 115 - 120 .