Integration of Robot Operating System and Ptolemy for Design of Real-Time Multi-robots Environments

Luis Costa, Alisson Brito, Tiago Nascimento, Thiago Bezerra

Integration of Robot Operating System and Ptolemy for Design of Real-time Multi-Robots Environments

Luis Feliphe S. Costa, Alisson V. Brito, Tiago P. Nascimento, and Thiago Henrique M. Bezerra

Federal University of Paraiba (UFPB)
Joao Pessoa, Brazil
luis.feliphe@dce.ufpb.br,alisson@ci.ufpb.br,tiagopn@ci.ufpb.br
http://laser.ci.ufpb.br

Abstract This work presents a technique for designing of real-time systems embedded applied to multi-robots scenarios, exploiting High-Level Architecture (HLA) standard for interoperability and simulation platforms, and Robot Operating System (ROS) with Robot-in-the-Loop simulations. The goal is to integrate existing consolidated standards and tools, rather than separately design and later hardly integrate. Through HLA, heterogeneous simulations can be synchronously co-simulated, enabling the joint execution of consolidated embedded system design tools for dedicated tasks, like, network simulation, circuits and algorithms design, etc. In parallel, ROS simplifies the task of creating complex robots across a wide variety of platforms. A bridge has been developed to provide an interface among HLA and ROS, exchanging data and keeping both synchronized. As proof of concept, the Ptolemy framework for Embedded System design has been integrated to Stage, a ROS compatible robotic simulator for 2D environments. This innovative integration has been successfully developed and validated, which enables future generalizations and opens opportunities to co-simulation of diverse tools for designing of embedded and robotics systems.

Keywords: Co-simulation, Robotics, embedded systems, Real-time systems

1 Introduction

Robotics is a multidisciplinary field, which demands cooperation of expertises and technologies from different research fields, i.e., Mechanics, Electronics, Artificial Intelligence, Embedded Systems. Each one brings consolidated methodologies and technologies that must perfectly work together.

Integration of different design tools, simulators and physical devices are not an easy task. Different efforts mainly formed by large communities of industrial and academic partners have successfully developed standards and tools to fill this gap.

When Embedded System Design is applied to Robotics normally gather different components with heterogeneous Models of Computation (MoC). Thus, tools with higher abstraction power are necessary in order to model, simulate and test all such MoCs, e.g., Finite State Machines (FSM), Synchronous Data Flow (SDF), Discrete Events (DE) and Continuous Time (CT). The Ptolemy II framework [1] is an example of a simulation and modeling tool intended to support system designs that combine different MoCs.

In 2007, a group of scientists, industries and engineers created an open-source robotic framework called Robot Operating System (ROS) [2]. It is a flexible framework for designing robots, providing a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. There are many other initiatives for merging different robot platforms [3].

Even more, high performance, availability and reliability of robots have turn them into increasingly complex computing systems. Dealing with such complexity requires most effective designing approaches, for example, using Hardware-in-the-loop (HIL) simulation, which means to add the real device in the simulation loop. This can enhance the quality of testing, reduce design time and anticipate usability tests immersed in the environment. Although its relevance, combine specific architecture designs and protocols with design tools is always a hard task. Some approaches, like in [4], present some methods to assist designers with hardware and simulation integration.

Thus, how would be possible to co-simulate different design standards and tools to form a unique simulation platform for designing of multi-robots scenarios? Aiming at filling this lack of tools and techniques, we have developed a simulation platform that combines Ptolemy (for Embedded System design), ROS (for robot design and real robots) and HLA (for synchronization and coupling of heterogeneous simulators). As proof of concept, a 2D robotic simulator called Stage was integrated to Ptolemy. A real robot also was integrated to provide an Robot-in-the-loop (RiL) simulation, this kind of simulation consists in using real robot in simulations to provide more realistic results. It is useful when it is not possible simulate an specific environment or when there is not enough robots to perform some experiment [5].

The remaining of this paper is organized as follows. Related works is presented in section 2, the proposed architecture is presented in section 3, while the experimental results are presented in section 4. Finally, section 5 tackles the main discussions and concluding remarks.

2 Related works

This work uses co-simulation environment where is possible test embedded systems and robotics. Works that perform only co-simulations of homogeneous models often are unable to implement the heterogeneity due to the great effort that is employed for adaptation to different hardware platforms, communication and synchronization protocols. Such characteristics like heterogeneity, synchroniza-

tion and testing with different Models of Computation are provided in this work through the integration of Ptolemy and ROS through HLA standard.

In [6] is presented the integration of heterogeneous simulators through HLA. A framework was developed for rapid integration of different simulators. A car-to-car communication application was presented, were SystemC is used to model the electronic controller of a car, and Omnet++ and Sumo were used to simulate network communication and car traffic, respectively. Present work also will provide interoperation between simulators using HLA, however to integrate Stage and Ptolemy.

An Architecture for Robot-in-the-loop is present in [7]. It proposes to separate sensors/actuators from decisions models of robots using an interface that allows different kinds of sensors to decision models. Also it cites the mainly steps to develop simulations: to develop a conventional simulation in which all robots are simulated; to develop an mixed environment with simulated and real robots; and make an experiment with real system were all real robots execute on real environment and have real sensors and actuators. We create a similar environment, where developer choose which sensors/actuator want to use.

In citeartigoExemploTresRiL is proposed a continuation of [7]. It is presented three examples of how Hardware-in-the-loop can be use on different situations. It uses an Management Model that is parted in two: Time manager and space manager. In first example, one robot was used to avoid obstacles, the second one evolves robots formation with two robots where each robot try find other one. In this experiment robots have not interconnections between themselves. When they meet each other, an software named manager running on laptop creates an connection between themselves and they work like a team. On experiment one robot is leader and the other one is an follower. One is real and other is simulated. In third scenario, an robotic patrol is used starting with an undefined number of robots (n>1). These robots stay on line, each one with two neighborhoods: one forward and other backward. This experiment shows that is possible to use more than one real robot on RiL simulation. This system does not use global coordination. We develop a similar experiment, but to make a real robot avoid simulated obstacles on Stage simulator, providing a Hardware-in-the-loop Simulation.

This work represents a considerable advance compared to the last works, were multiple instances of Ptolemy were integrated to performance improvement [8] and, specially, hardware devices were integrated to Ptolemy for verification [9].

3 Architecture

The proposed architecture can be seen in Fig. 1. The environment is divided in two parts, the first one is the ROS environment (in yellow) with ROS Robot Nodes, the Bridge ROS Interface, and ROS Core, which is responsible for communication between ROS nodes. The second part is the HLA environment (in blue) with the RTI, the Bridge Ambassador and the Ptolemy Ambassador.

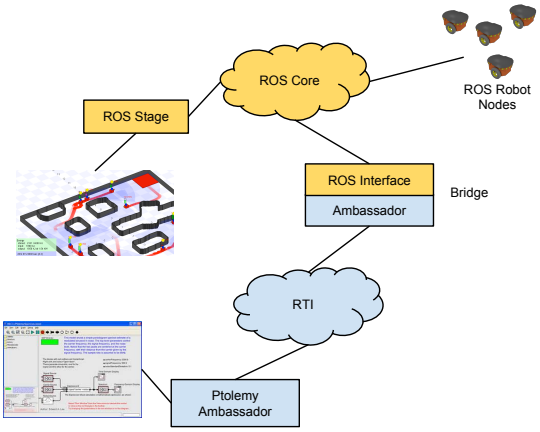


Figure 1: Proposed architecture

The intersection point between these environments is the Bridge. It is possible because the Bridge is an ROS Node and also implements an Ambassador. To became a ROS node, bridge uses Rospy library [10] that allows a Python application subscribe and publish on ROS topics. The ROS topics are responsible to share an specific information from ROS environment (e.g., speed, position, data from sensors, etc). The Ambassador is an interface with methods used by HLA to share information and manage time simulation, it was developed using Pyhla library [11] . Thus, it make possible to have multiple robots in ROS sharing their data with any others simulators that implements an HLA Ambassador, like Ptolemy.

3.1 Robot description

In HLA an object-oriented paradigm is used to describe data, called Federate Object Model (FOM). There, it is possible to describe classes, objects, attributes and hierarchy of classes. Once configured, the Bridge maps all necessary ROS Topics and attributes to objects and members of HLA. Both ROS and HLA may use the publish-subscribe protocol, the Brigde must just know for each subscribed variable from one side, to which variable it must publish in the other side.

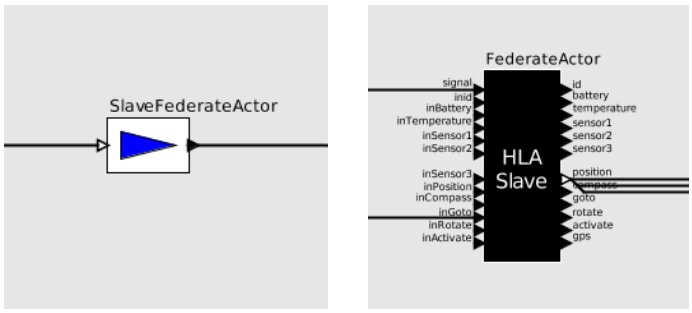
The description of these variables is presented following. For more details about Federation Object Model (FOM) rules and syntax, refer to [12].

```
(FED
  (Federation TestFed)
  (FEDversion v1.3)
  (spaces)
```

```
  (objects
    (class ObjectRoot
      (attribute privilegeToDelete reliable timestamp)
      (class RTIprivate)
      (class robot
        (attribute id reliable timestamp)
        (attribute battery reliable timestamp)
        (attribute temperature reliable timestamp)
        (attribute sensor1 reliable timestamp)
(attribute sensor2 reliable timestamp)
(attribute sensor3 reliable timestamp)
(attribute position reliable timestamp)
(attribute compass reliable timestamp)
(attribute goto reliable timestamp)
(attribute rotate reliable timestamp)
(attribute activate reliable timestamp))))
  (interactions))
```

With conclusion of FED file, an Ptolemy Actor was necessary to provide communication with HLA using this FED file, making possible to Ptolemy create or join on federation. Previous works already develop this kind of actor [13] [14], so they were modified to work with new FOM.

The Slave Federate actor was changed to manage new variables of fed file. The figure 2 shows old Slave Federate Actor and the new one.



(a) Old actor (b) New actor

Figure 2: Actors that allows communication between Ptolemy and HLA

The Ptolemy Actors, like the bridge, implements an Ambassador. This make possible to models from Ptolemy communicate with High Level Architecture, sharing data and managing time simulation. To change the Object Model and allows communication with stage, new ports were used to each specific information from fed file. To send and receive data the inputs and outputs ports were used respectively.

3.2 General architecture

According to HLA specification, individual simulators, called Federates, interoperate with each other through a Runtime Infrastructure (RTI), forming a Federation. The RTI is responsible for data exchanging and synchronization. Each Federate must implement an Ambassador, which translates the data and timing policies of the specific simulator to a common standard used by all other Federates. In this one we have used the implementation of a project named CERTI [15].

As it can be seen in Figure 3, two federates were used. The Ptolemy Federate, which provides the interface among RTI and Ptolemy. On the other side is the ROS Federate that interfaces with ROS environment.

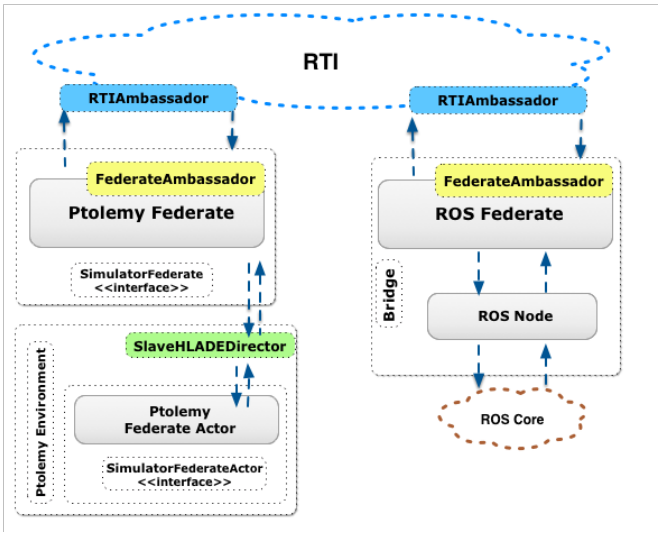


Figure 3: General architecture

In the Ptolemy side, a specific Director is responsible for coordinate the simulation, send data to RTI when necessary and, when data arrives from there, send them to specific actors also developed in this project. As proof of concept we have integrated the Stage Robot Simulator (http://rtv.github.io/Stage), Ptolemy and a real robot.

4 Methodology

To develop this environment, an Model was created on Ptolemy. The proposed model must to contain the Slave Federate Actor to interact with federation and

other actors to process data from robot and generate commands to control the robot. These commands also are send to ROS using the Slave Federate Actor. The model uses an DE Director, an extension of Discrete Events (DE) Director from Ptolemy. It is responsible to manage the time in the the Ptolemy simulation and to communicate with Ptolemy Ambassador.

An Algorithm to control the robots was developed using Ptolemy. It receive data from distance sensor of avatar robot and send command to avatar and real robot go on, stop or go on slowly to wait mobile obstacles out of way. The algorithm was created using the Python Actor, that allow Ptolemy use Python Applications.

To use the Stage simulator, an world file is necessary. This file have information of starting position of all robots and objects on Simulator. It was configured to have three robots used as obstacles on simulations, one additional robot that represent a real robot and a wall. A Real Turtlebot also is used on simulation to compare behavior with the simulated robot. This make the simulation became an Hardware-in-the-Loop simulation.

Some other applications are used to perform the environment. The Runtime Infrastructure Gateway (RTIG), that is used to manage simulation and exchange of messages and a python application that is an ROS node and is responsible to control the obstacles robots.

4.1 Proof of concept

An experiment using the proposed tools was perform as proof of concept. Stage simulator was responsible to simulate the environment with the robots and obstacles, and Ptolemy for simulate the control algorithm for avatar and real robot. With this environment possible to create an environment for co-simulation with Robot-in-the-loop to increase reality.

Figure 4 presents how components are communicating between themselves. Blue side is composed by three components: the bridge, Ptolemy simulator running control algorithm, and RTIG. On green side, four robots are simulated on Stage simulator, where three are virtual robots and the avatar.

On Stage simulator, three virtual robots are used as obstacles to avatar robot. They do the same path going from left to right side crossing way of avatar robot. Avatar robot has to pass by the robots without collide until it arrives at the wall. Figure 5 shows the disposition of the robots on map, where the black square is the avatar robot and the other squares are the obstacle robots.

On simulation, the virtual robots on Stage, walk right side for left side crossing the avatar way. The Avatar must to go to wall and stop before clash with it. The Ptolemy Control Algorithm will make Avatar stop walk or walk slowly when the distance sensor of avatar detect some obstacle. When the obstacle move out of range, avatar back to walk normally. Then Ptolemy subscribes the distance sensor of avatar and publish linear and angular velocities on avatar and real robot.

The model used by Ptolemy present on Figure 6. Is possible see many components, the clock is responsible to generate events that are used by other actors,

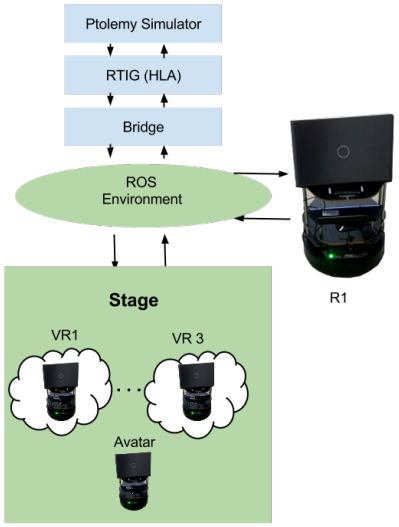


Figure 4: Experiment

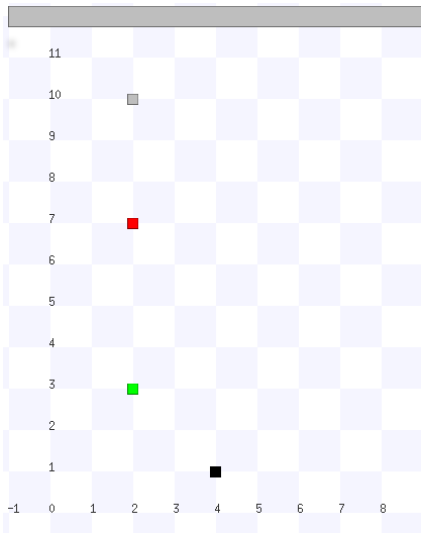


Figure 5: Disposition of the robots on map

the HLA Actor receive information from RTI and send it to output ports, the inputs values from HLA Actor are published to RTI. Control is the Python Actor with algorithm that receives data from sensor, and generate output converted into string and send to input of HLA Actor.

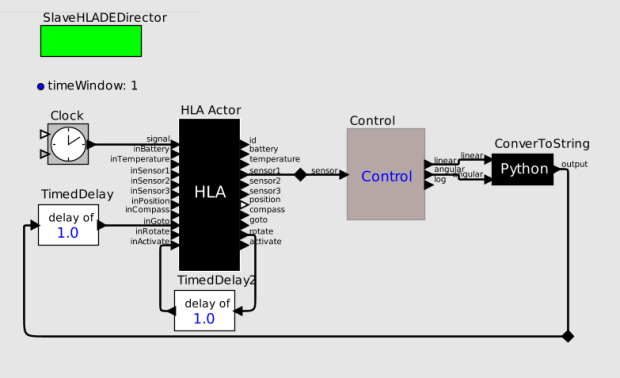


Figure 6: Ptolemy model

4.2 Results

The proof of concept was perform. The Ptolemy Simulator received information from robotic environment (Stage Simulator), and generate outputs to move real robot and avatar avoiding virtual obstacles. Both, avatar and real robot stops to avoid the obstacles during simulation. When obstacles moves out range, they back to walk.

One Hardware-in-the-Loop benefit was detected on experiment. During simulation was detected that Ptolemy was using too much velocity and was necessary decrease it to avoid bumps. Using only simulator for development process it would not be possible to detect this kind of problem.

5 Conclusions and Future Works

This work presents a technique for designing of real-time embedded systems in multi-robots scenarios, exploiting High-Level Architecture (HLA) standard for interoperability and Robot-in-the-Loop simulation, and Robot Operating System (ROS) framework for inter-operation among real robots and simulations. Then, a Bridge was developed to interface HLA and ROS components, or Federate Ambassadors and ROS Nodes. Also, a unique data model was defined to represent robot ports and attributes, and new actors in Ptolemy to send/receive data to/from these ports and attributes were developed.

The presented technique make possible perform experiments with many robots merging virtual and real robots. A benefit is when many robots are necessary to perform a simulation but just one is available. So, the real robot can be used to check characteristics from real world and the simulated robots can interact with the real robot. Also is useful when an environment is hard to create on real world, then it can be simulated.

Furthermore, other specific simulators could also be integrated and more detailed outputs could be collected, like power consumption, data traffic in the

network over different technologies and protocols, electro-mechanical and thermal analysis, etc. These possibilities will be explored in future works.

References

1. C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. [Online]. Available: http://ptolemy.org/books/Systems
2. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," *ICRA workshop on open source software*, vol. 3, no. 3.2, p. 5, 2009.
3. J. Kerr and K. Nickels, "Robot operating systems: Bridging the gap between human and robot," in *System Theory (SSST), 2012 44th Southeastern Symposium on*, March 2012, pp. 99–104.
4. M. Bacic, "On hardware-in-the-loop simulation," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Dec 2005, pp. 3194–3198.
5. X. Hu and B. P. Zeigler, "Measuring cooperative robotic systems using simulation-based virtual environment," in *Performance Metrics for Intelligent Systems Workshop(PerMIS)*, Aug. 2004.
6. C. Roth, H. Bucher, A. Brito, O. Sander, and J. Becker, "A simulation tool chain for investigating future v2x-based automotive e/e architectures," vol. 1, no. 1, February 2014, pp. 1739–1748.
7. X. Hu, "Applying robot-in-the-loop-simulation to mobile robot systems," in *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, July 2005, pp. 506–513.
8. A. V. Brito, A. L. V. Negreiros, C. Roth, and O. Sander, "Development and evaluation of distributed simulation of embedded systems using ptolemy and hla," in *17th IEEE / ACM International Symposium on Distributed Simulation and Real Time Applications*, 2013.
9. J. C. V. S. Júnior, A. V. Brito, , and T. P. Nascimento, "Verification of embedded system designs through hardware-software co-simulation," *International Journal of Information and Electronics Engineering*, vol. 5, no. 1, 2015. [Online]. Available: http://www.ijiee.org/index.php?m=content&c=index&a=show&catid=49&id=548
10. ROS. (2014, Nov.) Rospy wiki. [Online]. Available: http://wiki.ros.org/rospy
11. Nongnu. (2014, Nov.) Pyhla — python bindings for m&s hla. [Online]. Available: http://www.nongnu.org/certi/PyHLA/
12. IEEE, "Ieee standard for modeling and simulation – high level architecture (hla) – federate interface specification," *IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000)*, pp. 1–378, 2010.
13. A. Brito, A. Negreiros, C. Roth, O. Sander, and J. Becker, "Development and evaluation of distributed simulation of embedded systems using ptolemy and hla," in *Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on*, Oct 2013, pp. 189–196.
14. A. L. V. de Negreiros and A. V. Brito, "Analise da aplicacao de simulacao distribuida no projeto de sistemas embarcados," in *Simposio Brasileiro de Sistemas de Informacao*, 2013.
15. Savannah. (2014, Nov.) Certi resumo. [Online]. Available: http://savannah.nongnu.org/projects/certi