ARGoS-Blockchain interface

Volker Strobel Alex Pacheco Marco Dorigo

July 8, 2020

1 Introduction

Several recent works have addressed the combination of blockchain technology and swarm robotics. This combinatin proves useful for security, consensus finding, robot economies, robot-as-a-service, and potentially many more.

However, running these experiments requires a suitable framework. Such a framework proves useful both for researchers who do not have access to a suitable robot platform, such as the Pi-puck robot. Additionally, a suitable platform is important before running time- and cost-consuming experiments with real robots.

The ARGoS robot simulator (**?**) is the state-of-the-art research platform to conduct simulations in swarm robotics.

In general, in swarm robotics research, each robot is a blockchain node, and contributes to the maintenance of the system by exchanges blockchain information. To link ARGoS and blockchain software, we developed the ARGoS-Blockchain interface that provides access to the blockchain nodes for the robots (Figure 1). The interface is intended to facilitate research in blockchain-based robot swarms by allowing to call blockchain functions in ARGoS. Additionally, Docker makes it easy to install and run the interface on different platforms.

2 Software availability

3 Overview

The implementation of the custom Ethereum network is based on Capgemini AIE's Ethereum Docker[1]. Docker containers (**?**) contain all the necessary dependencies to run specific applications and are more lightweight than a virtual machine. In our setup, for each robot, the Ethereum implementation *geth* is executed in a separate Docker container. The simulated robots maintain a *custom* Ethereum network, i.e., a network that is shared among the simulated robots and independent of Ethereum's main network. Different containers can communicate with each other via channels.

[1] `https://github.com/Capgemini-AIE/ethereum-docker`, accessed on November 6, 2019

In order to execute an Ethereum function (e.g., create a new smart contract) from ARGoS, a robot uses its C++ controller to attach to the Docker container. The Docker containers provide shell scripts[2] with customizable templates (e.g., one of the templates compiles the smart contract, uses the binary code to send a blockchain transactions, and waits until the contract is mined). Via Ethereum's IPC (interprocess communications) interface, the shell scripts execute the Ethereum functions.

We use an auxiliary 'bootstrap' node for publishing the smart contract to the blockchain at the beginning of each run of the simulations (Figure **??**). The bootstrap node then mines the smart contract and sends the contract address and the ABI (application binary interface; the ABI specifies which functions a smart contract provides and how to call them) to the controllers of the robots. As soon as this is done, the bootstrap node is removed from the network. The bootstrap node is not necessarily required and the smart contract could also be created by a robot. However, we used an auxiliary node to make sure (i) that the smart contract is available at the start of the actual experimental run and (ii) that robots have the same initial conditions in all experiments.

The experiments were conducted on a computer cluster. To simulate the limited hardware of real robots, one core with 2.0 GHz and 1.8 GB of memory was assigned to each Docker container[3]. The communication channels between the Docker containers were only established when robots were within a 50 cm communication range in order to simulate the local communication capabilities of real robots.

The description in this report is intended to allow everyone to run blockchain-based robot experiments in simulation. There is no need for specific hardware but powerful CPU and RAM are advantageous for fluent experiments.

The ARGoS-Blockchain interface is composed of two modules.

3.0.1 Module 1: Blockchain part

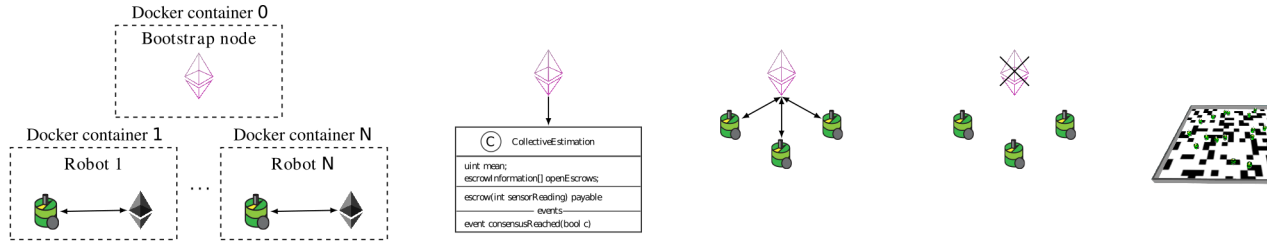
Module 1 allows for creating an Ethereum network with several nodes, where each node is located in a separate Docker container. For the ARGoS-Blockchain interface, the interaction with the Ethereum nodes is done via C++, using the code in the following repository: `https://github.com/Pold87/robot-swarms-need-blockchain`

This repository contains one module of the ARGoS-Blockchain interface that is described in the article "Blockchain Technology Secures Robot Swarms: A Comparison of Consensus Protocols and Their Resilience to Byzantine Robots by Strobel, V., Castello Ferrer, E., and Dorigo, M."

For debugging purposes or for creating your own private Ethereum network, you can also use this module without ARGoS.

[2]The interface uses shell scripts, since, during development, it became evident that they are executed much faster than other Ethereum APIs.

[3]This is a reasonable choice as a robot's computer could easily have such characteristics. It is also a convenient choice because on a computer with 2.0 GHz and 1.8 GB of RAM, Ethereum works "out-of-the-box," without any modifications; therefore, any interested user can obtain the most recent release of Ethereum from the official depository and use it with our publicly available ARGoS-Blockchain interface.



(a) One Docker container for the bootstrap node and N Docker containers for the robots are created. The Ethereum implementation geth is initialized with a custom genesis block, started, and accounts are created and unlocked for all nodes.

(b) The bootstrap node creates the smart contract and mines the first blocks of a custom Ethereum network to include the contract in the blockchain.

(c) The robots (i.e., the Ethereum software that is started for each robot) connect to the bootstrap node and receive the first blocks as well as the smart contract address and ABI.

(d) The bootstrap node is stopped in order not to interfere with the experiment.

(e) The experime

Figure 1: This scheme gives an overview of the workflow of the ARGoS-Blockchain interface

3.0.2 Module 2: ARGoS part

4 Setup

The very first time you run this code, it is required to create the Docker image for the Ethereum nodes and initialize Docker Swarm as follows:

```
cd geth/
docker build -t mygeth .
docker swarm init
```

Additionally, you have to set the variable `DOCKERFOLDER` in the file `global_config.sh` to the full path where this repository is located on your computer, for example:

```
/Users/vstrobel/Documents/ARGoS-Blockchain-interface/
```

In order to be able to mine, you need to create the DAG datasets as follows (the creates files require approximately 2 GB disk space and the execution of the script can take take several minutes):

```
cd local_scripts/
bash create_dag.sh
```

5 Run

Usually, the network is created when a swarm robotics experiment is started, using one of the start scripts in `https://github.com/Pold87/robot-swarms-need-blockchain`.

However, you can also start the Ethereum network without ARGoS, using the following command:

```
bash local_scripts/start_network.sh <number of nodes>
```

That is, `bash local_scripts/start_network.sh 5`, would create a private Ethereum network with 5 nodes.

Acknowledgements

Volker Strobel and Marco Dorigo acknowledge support from the Belgian F.R.S.-FNRS, of which they are a Research Fellow and a Research Director respectively. Alexandre Pacheco acknowledges support from the Faculty of Applied Sciences of the Université Libre de Bruxelles.