ARGoS-Blockchain interface

Volker Strobel Alex Pacheco Marco Dorigo

September 1, 2020

Abstract

This technical report documents the installation and use of the ARGoS-Blockchain interface. This interface allows for conducting experiments in the robot swarm simulator ARGoS and the smart contract blockchain framework Ethereum. The blockchain part uses the container software Docker in order to facilitate the installation and the parallelization of the blockchain nodes. The use of blockchain-based smart contracts provides a framework for managing security issues in robot swarms. The interface is a convenient way to prototype blockchain-based control software in robot swarms. Byzantine robots, that is, robots that behave in an unintended way can disrupt the swarm behavior. In this technical report, we detail the motivation, setup, and usage of the ARGoS-blockchain interface.

1 Introduction

Several recent works have addressed the combination of blockchain technology and swarm robotics (4; 5; 3). This combination proves useful for security, consensus finding, robot economies, robot-as-a-service, and potentially many other applications in robot swarms. However, conducting experiments involving robot swarms and blockchain technology requires a suitable testbed. A widely used framework consists of the Pi-puck robot platform running the Ethereum blockchain network. However, not every researcher has access to physical robots, obtaining and maintaining them be both time-intense and cost-intense, not to mention the lengthy execution of experiments. Therefore, a simulation testbed is needed to quickly try and test different controllers and scenarios. The ARGoS robot simulator (2) is the state-of-the-art research platform to conduct simulations in swarm robotics. 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.

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 3). The interface is intended to facilitate research in blockchain-based

```
│ geth/
├── img/
├── local_scripts/
├── .gitignore
├── README
└── docker-compose.yml
```

Figure 1: The folder structure of the package.

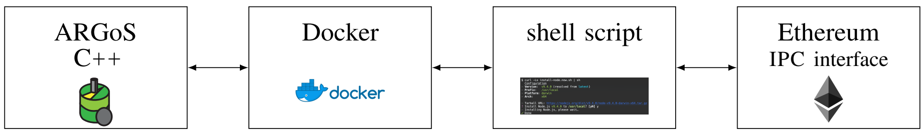


Figure 2: For each robot, the ARGoS-Blockchain interface establishes a connection to the Ethereum blockchain via a Docker container and shell scripts that provide templates for executing Ethereum (geth) functions.

robot swarms by allowing to call blockchain functions in ARGoS. Additionally, using Docker makes it easy to install and run the interface on different platforms.

2 Software availability

The open-source software is available at GitHub: the blockchain module is available at [1] and the ARGoS module is available at [2].

3 Overview

The implementation of the custom Ethereum network is based on Capgemini AIE's Ethereum Docker[3]. Docker containers (1) 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.

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

[1] `https://github.com/Pold87/ARGoS-Blockchain-interface`
[2] `https://github.com/Pold87/robot-swarms-need-blockchain`
[3] `https://github.com/Capgemini-AIE/ethereum-docker`, accessed on November 6, 2019

container. The Docker containers provide shell scripts[4] 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[5]. 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

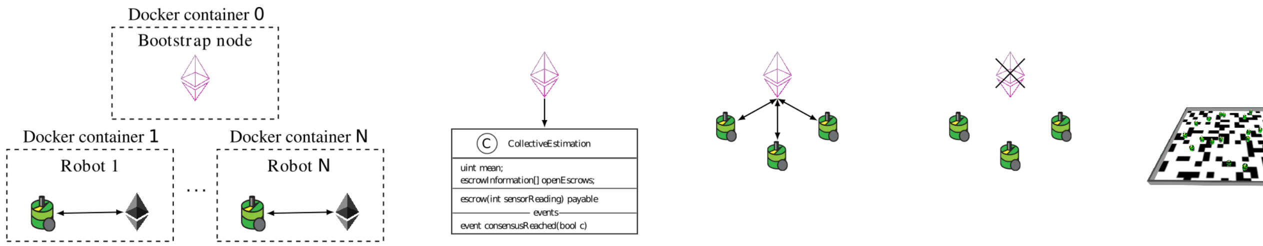
Module 1 allows for creating an Ethereum network with several nodes, where each node is located in a separate Docker container. For the ARGoS-Blockchain interface, the interaction with the Ethereum nodes is done via C++, using the code in the following repository: `https://github.com/Pold87/robot-swarms-need-blockchain`

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.

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

[5]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 3: This scheme gives an overview of the workflow of the ARGoS-Blockchain interface

3.0.2 Module 2: ARGoS part

4 Setup

The following setup steps describe the installation using Linux as operating system.

4.1 Requirements

- Sufficient disk space (at least 16 GB)
- Docker
- ARGoS[6] and the ARGoS e-puck robot plugin[7] (carefully read the instructions on the respective respositories)

4.2 Installation — Module 1: Blockchain module

First, obtain the source code for the blockchain module:

```
$ git clone https://github.com/Pold87/ARGoS-Blockchain-interface
```

Then, you need to create the Docker image for the Ethereum nodes and initialize Docker Swarm as follows. By default, the execution of Docker commands requires root user privileges. To facilitate the execution of Docker commands, it is possible to run Docker as non-root user (i.e., without `sudo`) as follows (see `https://docs.docker.com/engine/install/linux-postinstall/` for more detailed instructions):

[6] `https://github.com/ilpincy/argos3`
[7] `https://github.com/demiurge-project/argos3-epuck`

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

Then, create the Ethereum image.

```
$ cd ARGoS-Blockchain-interface/ && cd geth/
$ docker build -t mygeth .
$ docker swarm init
```

TODO: Run script with NUMROBOTS=0
TODO: Changer from python to python3 in Dockerfile
TODO: Download geth and put it into myowngeth/
TODO: Set the ARGoS folder in start$_network.sh$

4.3 Installation: ARGoS module

```
git clone https://github.com/Pold87/robot-swarms-need-blockchain
cd robot-swarms-need-blockchain/
mkdir build/
cd build/
cmake ..
make
```

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

```
DOCKERBASE='/home/vstrobel/Documents/docker-geth-network/'
```

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.

6 Debugging

First, check that the Docker container for the bootstrap node and the individual containers for the robots are running:

7 Usage

The functions of the interface can be found in the file `/interface/generic`$_interface.cpp$.

- `startMining: Start the mining process using one thread.`
- `stopMining: Stop the mining process.`
- `addPeer: Add a peer based on its enode.`

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.

References

[1] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), 2014.

[2] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271--295, 2012. version 3.0.0-beta48.

[3] V. Strobel, E. Castelló Ferrer, and M. Dorigo. Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. Technical Report TR/IRIDIA/2017-013, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, April 2018.

[4] Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In M. Dastani, G. Sukthankar, E. André, and S. Koenig,

editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, pages 541--549, Richland, SC, USA, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Volker Strobel and Marco Dorigo. Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation. In Marco Dorigo, Mauro Birattari, Christian Blum, Anders L. Christensen, Andreagiovanni Reina, and Vito Trianni, editors, *Swarm Intelligence -- Proceedings of ANTS 2018 -- Eleventh International Conference*, volume 11172 of *Lecture Notes in Computer Science*, pages 425--426, Cham, Switzerland, 2018. Springer.