Blockchain-based Multi-Robot Path Planning

Amr Mokhtar [1]
School of Electronic Engineering
Dublin City University
Dublin, Ireland
amr.mokhtar2@mail.dcu.ie

Noel Murphy [2]
School of Electronic Engineering
Dublin City University
Dublin, Ireland
noel.murphy@dcu.ie

Jennifer Bruton [2]
School of Electronic Engineering
Dublin City University
Dublin, Ireland
jennifer.bruton@dcu.ie

***Abstract*—The blockchain is a secure and trustworthy distributed transaction management system that is being extensively researched and developed for various applications and use cases. This study introduces a novel distributed control system using the Blockchain. A multi-robot path planning application is developed and deployed to benchmark a blockchain platform, the Hyperledger Fabric. Blockchain technology has the reputation of being sufficiently dilatory that it is inappropriate for time-sensitive applications. This research demonstrates how the enterprise-grade blockchain solutions overcome this shortcoming and investigates their potential for enabling secure and trusted distributed control systems for IoT.**

Keywords—Blockchain, Robotics, Distributed Computing, Multi-robot, Path Planning, Internet of Things

I. INTRODUCTION

Path planning for robots is the process of calculating a route that a robot takes traversing from point A to point B without interfering with other objects along the way. *Multi-robot path planning* is when this task is being undertaken collaboratively by a robot team. The *workspace* is the space where the robots operate with a population of objects and other robots. The multi-robot path planning domain focuses on solving collision-avoidance and motion coordination problems for the purpose of obtaining a collision-free environment.

Motion coordination solutions vary based on the mission of the robot team and the distribution of planning information. They can be either *centralized* or *decentralized* [1]. The centralized approach shares global information for the robot team so that it acts as one composite system and computes their individual paths usually using one of the classical planning methods such as A* [2]. On the other hand, decentralized approaches enable individual robots to compute their own paths independently, then coordinate by either (1) by planning in a prioritized order [3] such that a robot considers the forerunners as moving obstacles, or (2) by adapting their velocities [4] so that the resulting trajectories are collision-free; i.e., robots cross the same point but at different times.

[1] A. Mokhtar is with Network Platform Group, Intel Corporation, Shannon, Ireland (e-mail: amr.mokhtar@intel.com).
[2] N. Murphy and J. Bruton are with Entwine Research Centre, Dublin City University, Dublin, Ireland.

Before the commencement of the path planning operation, a robot must probe its surrounding workspace in a process called *environment modelling* which concludes with a definition of where the obstacles reside, and in what space the robot can freely navigate. The *roadmap* is an environment modelling method that plots a network of connected lines (also called *edges*) such that start and goal points (also called *nodes*) can be connected by a continuous path. The roadmap falls wholly in the *free space* so that any selected path on the roadmap is guaranteed to be free of collisions.

A. Internet of Robotic Things

The Internet of Robotic Things (IoRT) [5] domain bridges the robotics domain with the Internet of Things (IoT), enabling robots to execute more complex and larger scale operations. IoRT integrates intelligent and autonomous devices into a distributed architecture of platforms operating in the cloud and at the network edge.

The constrained processing power and memory bandwidth of connected robots have motivated the rise of *Cloud Robotics* [6] concept which is about reducing the requirement for robotic computing power and storage by offloading slices of the computation up to the cloud and let it 'do the heavy lifting'.

B. Blockchain Technology

Bitcoin cryptocurrency was first introduced by Satoshi Nakamoto [7] as an online distributed peer-to-peer electronic cash system. This system is solely operated by a trustworthy distributed ledger technology called the *Blockchain*. The blockchain is a cryptographically-chained list of blocks of encapsulated and digitally-signed transactions that were agreed upon and easily verifiable by a community. A blockchain network is a peer-to-peer network where all users operate on their own replicas of the shared ledger. Users transact via private-public key pairs. The user is addressable on the network by its public key, while the private key is kept a secret. When a user performs a transaction, it is digitally signed using their private key. This digital signature proves that it came from the owner and prevents anybody else from altering the transaction. The cryptographic nature of this process

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

guarantees that it is effectively impossible to guess or reversely infer the private key from the digital signature.

C. Smart Contracts

Smart contracts are executable scripts that are stored on the blockchain and accordingly executed by all the peers on the network in the endeavor of fulfilling the terms of the contract. The smart contracts concept goes back to Nick Szabo [8] and recently got re-utilized in the state-of-the-art blockchain frameworks such as Ethereum [9] and Hyperledger [10].

Since the robotic swarms become interconnected and heterogeneous, they inherit the traditional privacy, security and trust issues of the IoT space [11]. Integrating them with the permission-less, public blockchain technology introduces a solution to these issues but at the cost of compute, power and latency inefficiency. On the other hand, the permissioned, enterprise-grade blockchain solutions adopt an architectural approach which is addressing those inefficiencies, allowing them to own the potential of providing innovative solutions to emergent issues of the swarm robotics and to promote their capabilities [12].

This research study introduces a time-sensitive distributed control system using an enterprise-grade smart contract based blockchain platform in the field of robotic path planning with appropriate consideration for time constraints, performance, and autonomy requirements. An area that was not sufficiently approached by the current academic research.

In the following sections, the proposed solution is explored, investigated and benchmarked. The solution is developed and integrated with the Hyperledger Fabric platform (v1.1.0) [13] and the source code is viewable in GitHub [14].

II. Multi-Robot Path Planning

The decentralized coordination approach is used here for the multi-robot path planning. Individual robots compute their own paths sequentially in a prioritized order based on the Probabilistic Road Map (PRM) algorithm [15]. All robots are given the workspace specification and the stationary obstacles. The PRM path planning mechanism involves two main stages: (a) roadmap buildup and (b) pathfinding.

A. Roadmap Buildup

The roadmap is constructed by a group of randomly generated nodes which fall in the free space of the given workspace. The nodes are interlinked with a set of nearby nodes forming another group of edges. The generated nodes are validated for coincidence with any of the existing stationary obstacles. Similarly, the edges are checked for intersection with any of the other collaborating robots' paths. If a node is found interfering with an obstacle, or an edge intersecting with other edges or obstacles, they are immediately excluded from the constructed roadmap.

B. Pathfinding

The roadmap buildup stage concludes with a collision-free set of interlinked nodes that are neither colliding with static obstacles nor with other paths, as shown in Fig. 1. Thereby, any continuous trajectory that is marched from a start (S) to a goal (G) point through those constructed edges would constitute a successful robot path.

III. Hyperledger Fabric

The Hyperledger Fabric [13] is a Linux Foundation open source project which provides an enterprise-grade permissioned blockchain platform that delivers an elastic and extensible architecture for distributed ledger solutions. A permissioned blockchain platform restricts entities from appending blocks to the blockchain and enforces clear identities for all transacting parties.

In distinction to the controversial Proof-of-Work (PoW) consensus mechanism in use by Bitcoin, Hyperledger Fabric adopts an alternative computational and power-efficient consensus mechanism called the *ordering service*. Ordering services are configurable based on the shape of the relationship between their participants. Hyperledger Fabric is set to support ordering services like SOLO, Kafka and Byzantine Fault Tolerance (BFT). The consensus mechanism is the collaborative process of keeping all participants in the network consistently synchronized.

Hyperledger Fabric is a smart-contract-based blockchain platform. The smart contract, or the *chaincode* in Fabric terminology, is a piece of software that runs on the distributed ledger (the data storage part of the blockchain) where it is capable of retrieving and constructing blocks into the blockchain. Chaincode supports an event service which enables the implementation of event-driven finite state machines in client applications that are transacting through the blockchain.

A. Network Configuration

The Hyperledger Fabric network used in this project is configured as shown in Fig. 2. Multiple Robot client applications are implemented using the Java Software Development Kit (SDK) for Hyperledger Fabric [16] and logically connected with one peer node (peer0.org1.dcu.ie). The peer node hosts the *multi-robot* chaincode and its own replica of the ledger. The ordering server node (orderer.dcu.ie) is responsible for encoding blocks and committing them on the

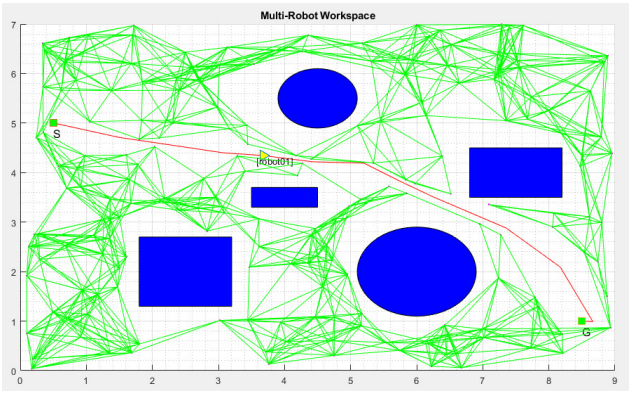

Fig. 1. Probabilistic roadmap for the second robot. The edges (green links) do not collide with the previous path (red line) or with the static obstacles (blue circles and rectangles).

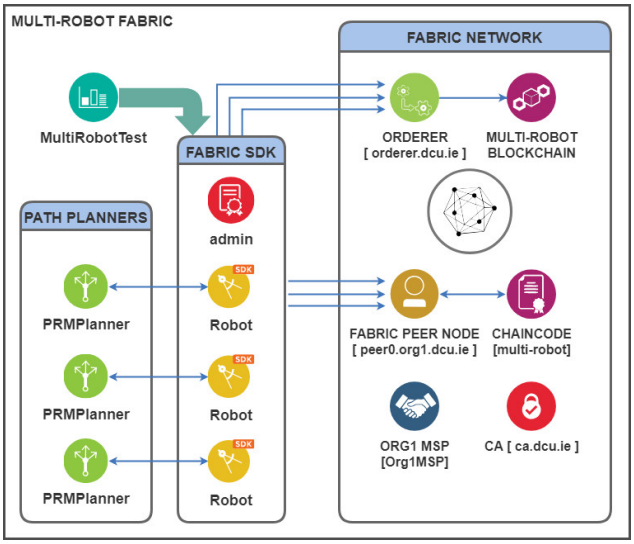

Fig. 2. Hyperledger Fabric network configuration. Multiple robot application built using Fabric SDK running on a multi-robot chaincode.

ledger. The Certificate Authority (CA) issues digital certificates that represent the digital identities of the participants on the Fabric network. An organization assigns a Membership Service Provider (MSP) that helps in identifying member parties of the organization. Every organization has its own MSP member; for org1 it is *Org1MSP*.

The consensus mechanism in use by this network configuration is the *SOLO* ordering which is a centralized ordering service. Although it supports batching of transactions into a single block, for this application it was configured to ensure commitment of single transactions into individual blocks to maintain the real-time delivery of transactions into the ledger.

The simulation program of the multi-robot path planning is the test application "MultiRobotTest" which is a high-level program that supervises the overall robot client applications. It allocates virtual instances of robot client applications that are interacting with each other through the Hyperledger Fabric network seeking a consensus on their collective path plans.

B. *Data Structure*

The ledger subsystem is comprised of: *the world state* and the *transaction log*. Chaincode invocations execute transactions against the world state (i.e. the current state data) – a more efficient means of data retrieval than strolling through the whole transaction log. Fabric ledger supports *CouchDB* and *LevelDB* data stores. LevelDB is a key-value store that is embedded by default in the Fabric and is the data store that is accessed by the multi-robot chaincode. The static workspace JSON representation is stored as the value and identified by the key "workspace." Whenever a path plan is set by a robot client application, it is similarly stored as a JSON value with the key set to the robot name. The robot name is a unique identifier that is used by the robot throughout all interactions in the network.

When a robot client application sends a transaction, it ends up invoking a method that is exposed by the chaincode. The main invocation methods provided by the multi-robot chaincode are:

- **getWorkspace**: this invocation expects no arguments from the Robot application. The LevelDB database is queried by the key "workspace" and accordingly returns the workspace configuration that was set when the chaincode was first instantiated. The workspace is not expected to be changed at runtime. If it is the case, then a new chaincode must be instantiated. Hyperledger Fabric supports upgrading chaincodes with newer versions on the same ledger.
- **setMyPath**: this invocation expects two arguments; the robot name and its computed robot path plan. A particular robot can have only one path plan allocated at one time. Therefore, subsequent calls to this method will effectively replace the allocated path plan for this robot with the new one.
- **getAllPaths**: this invocation expects one argument; the robot name and returns a list of all path plans allocated for the robotic teammates. This can only be performed by querying the ledger with the stored robot names first, then querying again using these names to retrieve their path plans. It is achieved by the use of *composite* and *partial composite* keys, which allows flexible and efficient ledger indexing. The composite key "allpaths~robotId" is used to query the database for the robot names that have path plans in the data store. Then, their path plans are retrieved by the use of the partial composite key "allpaths" which is passed to the chaincode API "GetStateByPartialCompositeKey."

C. *Robot Client Applications*

A robot client application is the piece of software that a physical robot should execute in a real-world scenario for the purpose of path planning. The robot client application enrolls to the Fabric network by registering itself and connecting to a Fabric peer (peer0.org1.dcu.ie). The Fabric peer node deploys the multi-robot chaincode and maintains a replica of the ledger. Robot client applications have their own PRM path planners that execute the PRM algorithm on the given workspace.

For the PRM path planning program to function, it requires knowledge about the workspace configuration and the path plans of all the other robots in the team. Thus, the robot client application boots up by querying the chaincode for the workspace configuration by invoking the *getWorkspace* method. The workspace definition is returned in JSON format that the robot client application parses and stores in its local memory for further processing. Similarly, other robots' path plans are queried through a *getAllPaths* method invocation. Thereafter, the PRM planner is called and commences building up the roadmap and generating its path plan. A new transaction proposal is constructed with the new path plan and the *setMyPath* method is invoked. The multi-robot chaincode executes against the current state of the ledger and provides a response value and a read/write set. The transaction goes through the endorsement policy and winds up at the ordering

service that creates a block for the transaction message and appends it to the ledger. Once committed, the new block is broadcasted to all peers and the *path-committed* event is emitted to all robot client applications. Now, the robot team is synchronized and in agreement on their individual path plans.

IV. Simulation Results

A Hyperledger Fabric v1.1.0 testing network was used in simulating and benchmarking the multi-robot path planning application covered by this paper. The purpose was to measure various metrics such as consensus time and latency. The Hyperledger Fabric network was simulated in an Amazon Elastic Compute Cloud t2.micro instance (1 vCPU and 1GB RAM) with Ubuntu Server 16.04.4 LTS operating system installed.

A. Multi-Robot Test Application

The multi-robot test application is designed to instantiate robot client applications in a prioritized order (prioritized planning [3]). One robot is given the go-ahead while the others wait. When the path plan is committed, the next one in the line 'steps up' and commences planning while considering all pre-computed path plans, and so on. This sequence iterates until a consensus is reached (all the computed path plans are free of collisions.)

The *Consensus Time* is the time elapsed from the first robot starting its planning until all robots have finished planning and the consensus is reached. This elapsed time is divided into two portions: the first portion is the PRM path planning, denoted as *Path Planning Time*. This is the time taken by a robot to execute the roadmap buildup and pathfinding. The second portion is the ledger commit time, the *Ledger Commit Latency*. This is the time consumed by a transaction proposal to get committed on the ledger.

The workspace configured in this benchmarking activity was comprised of three rectangles and two circles. The starting point was at (1,5) and the goal is set at (9,1).

The multi-robot test was executed for multiple scenarios, starting with 2 robots up to 8, synchronously transacting over the common Fabric network. Fig. 3 depicts the resultant path plans in the 8-robot scenario. Each robot application is configured to construct a roadmap of 1000 nodes, where each node interlinks 40 edges with its neighbors. It worth noting that selecting the overall number of nodes and the number of edges per node is a trade-off, and values must be chosen carefully. The larger the values, the greater the accuracy and success rates, but at the cost of increasing the execution cycles and computation time, and vice versa.

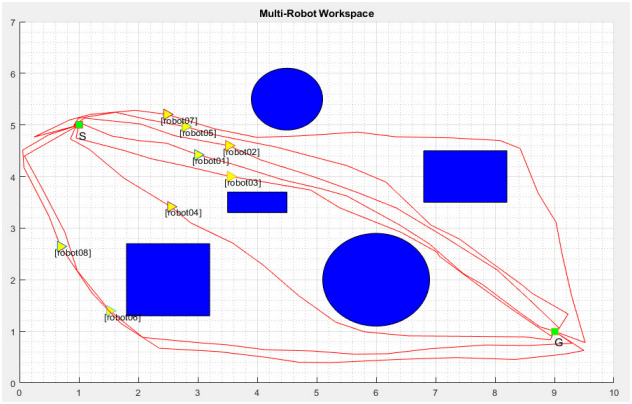

Fig. 3. Path planning over Hyperledger Fabric network for 8 robots.

B. Consensus and Planning Execution Times

The consensus and planning times of the prioritized path planning for multiple robots are captured and plotted side-by-side in Fig. 4 to visualize how they scale up together when the number of robots increases from 2 to 8, and to demonstrate what portion of the consensus time is taken up by the path planning task. The region between the two lines represents the total ledger commit latency. It is observed that the total path planning time consumes an average of 81.46% from the total consensus time. This is because PRM path planning is a compute-intensive task, particularly the roadmap buildup which generates a scattered number of nodes, interlinks them and executes the collision detection computations.

C. Ledger Commit Latency

Another benchmarking metric is the *Ledger Commit Latency*. Latency is a pivotal performance measurement, particularly for IoT applications. It is defined as the amount of time that a message takes to traverse a network. In this document, it is synonymous with the term *delay*. Thereby, the ledger commit latency is the delay from the point of sending a transaction proposal (which contains a computed path plan) on the Fabric network until the path is actually committed on the ledger.

The *Path Planning Latency* is a measurement that represents the execution time of the PRM path planning algorithm (roadmap buildup + pathfinding). It is the time taken from the robot client application being triggered until a path is computed and becomes ready for sharing on the Fabric network.

The path planning and ledger commit latencies are captured individually for each robot in the 8-robot prioritized path planning scenario and are depicted side-by-side in Fig. 5 to show the amount of latency being taken by path planning in

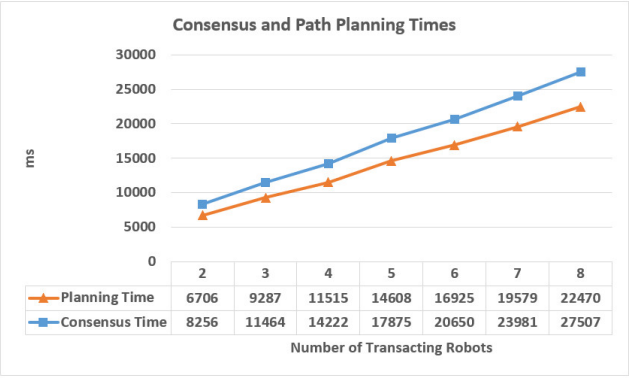

Fig. 4. Consensus and Planning Times for multi-robot scenarios. Time units are milli-seconds.

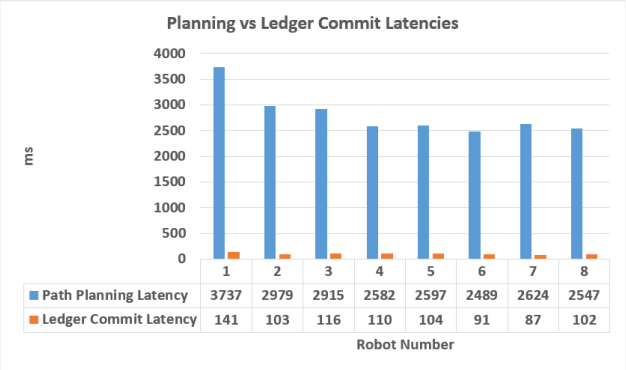

Fig. 5. Planning versus ledger commit latency times for 8-robot path planning scenario. Every column represents the latency incurred with every individual robot. Time units are milli-seconds.

contrast to ledger committing tasks. Calculating the average ledger commit latency results in approximately **112.46** ms. This is indicating that IoT applications can avail of the blockchain framework with a minimal overhead that is of the order of milliseconds as compared to approximately 10 minutes for Bitcoin and 12 seconds for Ethereum in order to confirm their transactions [17]. It worth noting that the aforementioned experiments are preliminary, they were performed on a basic computing system, and they have significant room of improvement. Using more capable compute platforms will scale the performance and reduce the latencies.

V. Discussion and Future Work

Prioritized path planning is a blocking mechanism that necessitates one robot planning at a time while the others are waiting. Adapting this approach to allow individual robots to work in parallel reduces the consensus latency. The roadmap buildup stage can execute independently and parallelly for every robot, but the pathfinding stage must be more interactive. In order to do that, the robots need to collaborate in smaller steps.

A robot's path is a set of line segments that are constructed from the edges of the PRM roadmap. Therefore, collaboration can be conducted at the level of line segments rather than computing the whole path in advance. This improves the collective planning quality and reduces the consensus latency. All robots would commence building the roadmaps in parallel, then interact on the smart contract in phases until the final target point is reached for all. The consensus must be achieved at every phase before moving to the next one. With this approach, the overall path planning time is effectively reduced to the time of a single robot. It worth mentioning here that from our experiments, it was observed that pathfinding overhead is too small as compared with the roadmap computation. Thereby, the computation overhead of the repetitive pathfinding tasks can be ignored and the path planning time will be considered as mainly driven by the roadmap computation.

Although the latter interactive approach is more efficient, it increases the transaction frequency on the smart contract and the ledger. Thus, this approach is heavily impacted by the ledger commit latency. As an example, assume that a team of 8 robots each has a path of 18 segments. So, for the best-case scenario, if a robot initiates one transaction per line segment, there will be 144 transactions committed to the ledger. For 112.46 ms latency per commit, the expected overall ledger commit time is approximately 16194.24 ms. Because of the path planning being mainly driven by the roadmap computation and that the roadmap was computed for one time by each robot, then the overall path planning time shall be considered as the path planning time of a single robot. Looking back to Fig. 4, the path planning time of a single robot is 2808.75 ms (dividing 22470 ms by 8). Thus, combining the previous results, the estimated consensus time = path planning time + ledger commit time results in approximately 19002.99 ms, which is more efficient than the 27507 ms consensus time of the prioritized approach.

Even though the interactive approach was estimated to outperform the prioritized approach, we have not analyzed how it will scale with larger path plans and larger numbers of robots. Also, further research and development towards optimizing the blockchain networks in terms of ledger commit latency will promote the interactive and coupled decision-making applications.

VI. Conclusion

In this paper, the investigation of a novel multi-robot path planning approach using the permissioned, enterprise-grade blockchain, Hyperledger Fabric platform, is described. The simulation results reveal minimal transactional latencies that are counted in terms of milliseconds, and thus much smaller than permission-less, public blockchain platforms like Bitcoin and Ethereum which have longer delays. The solution described shows considerable potential for enabling secure and scalable distributed control systems for IoT use cases. The use of smart contracts is a crucial facility for flexible and distributed execution of business logic that enforces the consensus achievement for the collaborative team workers.

References

[1] L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," Encyclopedia of Complexity and System Science, pp. 5783-5800, 2009.

[2] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, (2), pp. 100-107, 1968.

[3] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," Algorithmica, vol. 2, (1-4), pp. 477, 1987.

[4] K. Kant and S. W. Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition," The International Journal of Robotics Research, vol. 5, (3), pp. 72-89, 1986.

[5] P. P. Ray, "Internet of Robotic Things: Concept, Technologies, and Challenges," IEEE Access, vol. 4, pp. 9489-9500, 2016.

[6] G. Hu, W. P. Tay and Y. Wen, "Cloud robotics: architecture, challenges and applications," IEEE Network, vol. 26, (3), 2012.

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," [Online]. 2008. Available: https://bitcoin.org/bitcoin.pdf

[8] N. Szabo, "Formalizing and securing relationships on public networks," First Monday, vol. 2, (9), 1997.

[9] Ethereum Foundation. (2018). *Ethereum Project* [Online]. Available: https://www.ethereum.org/

[10] The Linux Foundation. (2018). *Hyperledger* [Online]. Available: https://www.hyperledger.org/

[11] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," Future Generation Computer Systems, vol. 82, pp. 395-411, 2018.

[12] E. C. Ferrer, "The blockchain: a new framework for robotic swarm systems," CoRR, vol. abs/1608.00695, 2016.

[13] The Linux Foundation. (2018). *Hyperledger Fabric* [Online]. Available: https://www.hyperledger.org/projects/fabric/

[14] A. Mokhtar. (2018). *Implementation of multi-robot path planning using Hyperledger Fabric platform* [Online]. Available: https://github.com/amr-mokhtar/multi-robot-fabric/

[15] L. E. Kavraki et al, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation, vol. 12, (4), pp. 566-580, 1996.

[16] Hyperledger Fabric. (2018). *Java SDK for Hyperledger Fabric* [Online]. Available: https://github.com/hyperledger/fabric-sdk-java/

[17] G. Hileman and M. Rauchs, "Global Blockchain Benchmarking Study," University of Cambridge, 2017.