Distributed Ledger Technology meets Distributed Shared Register Theory

Emmanuelle Anceaume, Romaric Ludinard, Maria Potop-Butucaru, Frédéric Tronel

Brief Announcement: Distributed Ledger Technology meets Distributed Shared Register Theory

Emmanuelle Anceaume[1], Romaric Ludinard[2], Maria Potop-Butucaru[3], and Frédéric Tronel[4]

1 **CNRS / IRISA, Campus de beaulieu, Rennes, France**
2 **ENSAI, Rennes, France**
3 **LIP6, Université P. & M. Curie, Paris, France**
4 **CentraleSupélec, Rennes, France**

Abstract

This paper introduces Distributed Ledger Register that mimics the behavior of most popular ledgers such as Bitcoin or Ethereum. Our work is the first to make the connection between the Distributed Ledger Register and the classical theory of shared registers. We furthermore, propose an algorithm that emulates the distributed ledger register.

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

In 2008, Satoshi Nakamoto, a pseudonymous author, published a white paper describing the Bitcoin network, a way to create, distribute and manage a currency that does not rely on a trusted third party [8]. Since then many crypto-currencies have been proposed, including the popular Ethereum [10]. In the following we detail the functioning of Bitcoin. Ethereum follows almost the same pattern with sligthly subtilities which are not relevant for our study. The Bitcoin network is a peer-to-peer payment network that relies on distributed algorithms and cryptographic functions to allow entities to pseudonymously buy goods with digital currencies called bitcoins. Bitcoin mainly relies on three types of data structures (i.e transactions, blocks and the distributed ledger – also called the blockchain) and three types of entities (i.e., user, Bitcoin node and miner) to offer such functionalities. A *transaction* allows *users* to transfer bitcoins from a set of input accounts to a set of output accounts. A *block* contains a list of transactions, a reference to its parent block (hence the name of *blockchain*), and a proof-of-work, that is a nonce such that the hash of the block matches a given target. We say that a block b is locally valid if it only contains locally valid transactions. *Bitcoin nodes* locally maintain a copy of the blockchain, and once validated, propagate newly transactions and blocks to all the entities of Bitcoin. Blocks are generated by *miners*, a subset of the Bitcoin nodes involved in the *proof-of-work* competition. This competition may result in multiple blocks referencing the very same parent block, and hence the creation of several chains. This situation is known as *blockchain fork*. Bitcoin defines the notion of *best chain* (the common history of the ledger on which miners agree), which corresponds to the longest chain starting from the genesis block of the distributed ledger (this is the unanimously agreed initial block of the distributed ledger). In the case of Etherium the best chain is the heaviest one. The *level of confirmation* of a block b belonging to the best chain of the distributed ledger is equal to the number of blocks included in the best chain starting from b. Nakamoto [8] has shown that if the proportion of malicious miners is $\leq 10\%$, then with probability $\leq 0.1\%$, a transaction can be rejected if its level of confirmation in a local

copy of the blockchain is less than 6. In case of Ethereum this level is 1. We say that a transaction is *deeply confirmed* once it reaches such a confirmation level.

Connection between the distributed computing theory and Bitcoin distributed ledger has been pionnered by Garay et *al* [4]. The main focus of the distributed community [1–6, 9] has so far been the distributed ledger agreement aspects. Our paper investigates consistency properties of the distributed ledger and makes the connection between the distributed ledgers and the distributed registers theory. In the following we propose the computational model, the register definition and the algorithm implementing the distributed ledger register.

2 Computing model

We consider a distributed system (Distributed Ledger system) composed of an arbitrary finite number of users, Bitcoin nodes and miners. We assume that all bitcoin nodes have capability to mine blocks, thus there are no distinctions between miners and Bitcoin nodes. We suppose the existence of a protocol managing the arrival, departure and connectivity of both users and miners in the distributed system. Both users and miners communicate by exchanging messages through reliable and authenticated channels. It is assumed that the system has a built-in communication abstraction, denoted broadcast, that provides both users and miners with an operation denoted broadcast(), and each miner with a matching operation denoted deliver(). Each entity in the distributed system (user and miner) is a state machine, enriched with the operations send and receive. Its state (called "local state") is defined by the current values of its local variables. A configuration (or global state) of the Distributed Ledger system (e.g Bitcoin and Ethereum) is composed of the local state of each entity in the system. The passage of time is measured by a fictional global clock. Users and miners do not have access at the fictional global time. At each time t, each entity is characterized by its local state. Both users and miners can suffer arbitrary failures. We assume that less than a fraction β (i.e. 50%) of the computational power of the system is owned by faulty miners. No such restrictions hold for faulty users.

3 Distributed Ledger Register

We now define a new type of R/W register, the multi-writer multi-reader register *Distributed Ledger Register*, $DLR(k)$ that mimics the behavior of Bitcoin and Etherium distributed ledgers. This register can be written by any miner and read by any user and miner. Parameter k represents the deep confirmation level of a block in the distributed ledger. Note that unlike the classical register definition [7], the $DLR(k)$ register operations span a complex data structure: $\mathcal{B}$ is an ordered sequence of blocks such that the first block and the last block of the sequence are the genesis and a leaf block of the distributed ledger respectively. Indeed, recall that in presence of forks, both Bitcoin or Ethereum distributed ledgers are trees of blocks and not a unique chain. A branch appears when a fork occurs. The value of a $DLR(k)$ register is a chain from the root of the tree to a leaf. The selection of the root is different from one ledger to another. In Bitcoin, this selection is based on the length (the longest chain will be conserved) while in Etherium the selection criterium is based on the chain weight.

Register $DLR(k)$ operations. Register $DLR(k)$ is equipped with a write and read operations: The $DLR(k)$.write($\mathcal{B}$) operation allows any writer to try to change the value of $DLR(k)$ with the input value $\mathcal{B}$. The $DLR(k)$.read() operation allows any reader to determine the value of $DLR(k)$. In order to specify the level of confirmation of a block, we introduce the notion of k-valid write. Operation $DLR(k)$.write($\mathcal{B}$) returns *true* if $DLR(k)$.write($\mathcal{B}$) is

k-valid otherwise it returns *abort*.

▶ **Definition 1** (k-valid)**.** Operation $DLR(k)$.write($\mathcal{B}$) is k-valid if and only if $\exists t, k > 0$ such that a virtual $DLR(k)$.read() invoked at time t after the invocation of $DLR(k)$.write($\mathcal{B}$) returns a chain $\mathcal{B}'$ such that $\mathcal{B}$=prefix($\mathcal{B}'$) and length($\mathcal{B}'$) $\geq$ length($\mathcal{B}$) $+ k$. Function length($\mathcal{B}$) returns the number of blocks that compose chain $\mathcal{B}$. Note that there is a non zero time between the invocation of the $DLR(k)$.write($\mathcal{B}$) operation and its return.

$DLR(k)$ **register specification.** A *DLR(k)* multi-reader multi-write register is defined by the following properties.

- *Liveness.* Any invocation of $DLR(k)$.write($\mathcal{B}$) or $DLR(k)$.read() terminates.
- *k-coherency.* Any $DLR(k)$.read() returns a value $\mathcal{B}$ such that $\mathcal{B}$ has a prefix $\mathcal{B}'$ where $\mathcal{B}'$ is the value of the register written by the last k-valid $DLR(k)$.write($\mathcal{B}'$) operation that happend before $DLR(k)$.read(). Note that for the first read it is assumed that at least one successful write operation happened before the start of the read.

▶ **Theorem 2.** *When $k > 0$ and k is bounded a DLR(k) register is equivalent to a regular register. When $k = \infty$ a DLR(k) register is equivalent to a safe register.*

▶ **Theorem 3.** *DLR-Algorithm implements a DLR(k) register.*

The DLR-Algorithm emulates the distributed ledger register. Each miner manages one local variable, called $\mathcal{TB}$, that stores the distributed ledger. This variable locally serves to construct the distributed ledger register.

- Function best_chain($\mathcal{TB}$) returns the best chain as explained above, and
- Function update_tree($\mathcal{TB}$, $\mathcal{B}$) fusions $\mathcal{TB}$ with $\mathcal{B}$.

```
Operation DLR.read () is % issued by a reader %
(01) return(best_chain(TB) )

Operation DLR.write (B) is % issued by a writer %
(02) update_tree(TB, B) ; broadcast (<propose B>)
(03) repeat B' = DLR.read ()
(04) until length(B') ≥ length(B) + k
(05) if B= prefix(B') return true
(06) else return abort

(07) upon deliver(<propose B>) update_tree(TB, B)
```

References

1 C. Cachin. Blockchain - From the Anarchy of Cryptocurrencies to the Enterprise (Keynote Abstract). In *Proc. of the OPODIS International Conference*, 2016.

2 C. Decker, J. Seidel, and R. Wattenhofer. Bitcoin Meets Strong Consistency. In *Proc. of the ICDCN International Conference*, 2016.

3 I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *Procs of the USENIX NSDI Symposium*, 2016.

4 J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Proc. of the EUROCRYPT International Conference*, 2015.

5 M. Herlihy and M. Moir. Blockchains and the logic of accountability: Keynote address. In *Proc. of the ACM/IEEE LICS Symposium*, 2016.

6 E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *Proc. of the USENIX Security Symposium*, 2016.

7 L. Lamport. On inter-process communications, part I: basic formalism and part II: algorithms. *Distributed Computing*, 1(2):77–101, 1986.

8 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

9 Pass R., Seeman L., and Shelat A. Analysis of the blockchain protocol in asynchronous networks. In *Proc. of the EUROCRYPT International Conference*, 2017.

10 G. Wood. Ethereum: A secure decentralised generalised transaction ledger. `http://gavwood.com/Paper.pdf`.