1 - Introduction

It is no surprise that late payments between companies are a big problem, and can seriously put at risk the stability and sometimes the survival of a company. Entrepreneurs who were running successful businesses had to file for bankruptcy because of lack of liquidity in the company's bank account, due to late or unsolved payments from customers. When entrepreneurs find themselves in these circumstances, they often have to borrow money in order to pay their bills, while waiting for their customer to pay them. This is a broken mechanism that puts several companies at the mercy of credit institutions, which can potentially determine the course of life of a company.

Debtloop addresses this problem by creating a global ledger which can be used to detect credit and debit loops between group of companies, and perform automatic settlement among them.

2 - Problem Definition

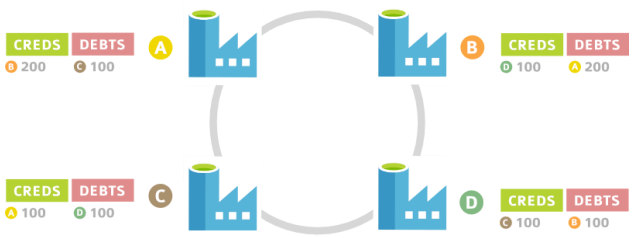
There are on the market several expense splitting mobile apps used to share bills within groups of friends. These apps, in addition to keeping track of all expenses in a group, automatically perform simple settlements. Given two persons A and B, for example, if A spends 25$ on behalf of B, and, later on, B spends 15$ on behalf of A, the app will perform all the required compensations and tell B to pay A only 10$.

This is a simple example, but the mechanism could work even better in complex scenarios where there are multiple parties involved. When a person A spends some money on behalf of a person B, it is because A trusts that B will pay back later. Also, today A is paying something on B's behalf, but tomorrow B might be paying something on A's behalf. With multiple transactions and multiple parties involved, it is very easy to see how debit and credit positions can compensate each other, reducing the circulating amount of cash between all parties.

If we bring the above example in a more complex business environment, a company A selling products to another company B, always issues an invoice to company B. Payment terms are always 30, 60 or even 90 days after the issuance of the invoice. At this particular moment, Company A trusts that Company B will pay the invoice later. At the same time, Company B sells product to another Company C, which will also pay with similar terms. Company C will then sell products to another company D, and so on.

However, all these companies, in order to sell products, also need to buy products or services. The result is that, at any particular moment, each company holds a series of debt and credit positions with other companies. Very often, in order to pay bills on time, a company is forced to borrow some money from a bank, which will issue a loan based on the credit positions that the company holds. This is a quite perverse mechanism, which unnecessarily increases the cost of money by introducing a trusted intermediary like a bank.

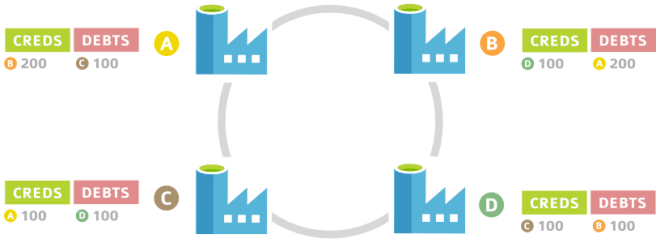
The drawing below depicts a common situation between four random companies A, B, C and D.



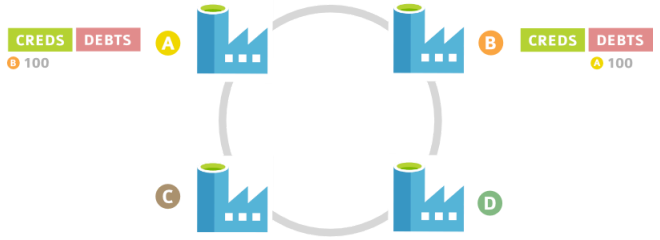
As it can be noticed from the drawing, Company A holds a debt position of $100 with Company C and a credit position of $200 with Company B. This means that Company A might need to borrow money from a bank in

order to pay its debts to Company C, while it is waiting for the payment from Company B. This mechanism can put in danger the economical situation of Company A and increase the cost of money.

However, if we look at the above scenario from a higher level perspective, the following situation might be revealed:



The debt and credit positions of Company A and B might be short-circuited by considering debt and credit positions of other companies. In the above example, Company C and D have open positions between themselves as well as Company A and B. This situation creates a loop between the positions of all four companies, as it is possible to see from the above drawing. It is easy to notice that an automatic settlement and debt and credit positions can be performed, resulting in a situation like the following:



Thanks to the detection of the loop most of the positions have been cleared; the only one remaining is a debt of $100 between company B and Company A. For simplicity, the above example only involves four companies. However, in a real situation a loop might involve tens or hundreds of companies, resulting in a much bigger benefit for the companies involved.

The problem today is that companies keep track of their balances in their own private ledger, making this loop detection impossible.

2 - High Level Solution

Debtloop tries to address this problem by creating an anonymous global ledger that offers companies the possibility of settling their debt and credit positions through the discovery of loops as described in the example above.

In order to operate *Debtloop* requires two type of users: *Bookkeepers* (BK) and *Loop-Breakers* (LB), plus the issuance of a token (*DEBT*) which is exchanged among the different types of users in the network.

Bookkeepers are usually companies registering debt and credit positions on the global ledger. All positions recorded are completely anonymous, so it is impossible for anyone on the network to know the real identity of the company. In order to register a position on the ledger, a *Bookkeeper* user keeps a minimum amount of *DEBTs* at stake in its wallet. Part of this amount is deducted when a loop is broken and an automatic settlement is performed by the network.

Loop-Breakers are users who crawl the global ledger in order to find loops and break them. Whenever a *Loop-breaker* breaks a loop, he is rewarded with an amount of *DEBTs* deducted from the wallets of all

Bookkeepers who are benefiting from the loop breaking and the consequent settling. In order to become a *Loop-Breaker*, a user must put a fixed amount of *DEBT* at stake in order to guarantee the credibility of the node.

3 - Solution Details

```
contract Position {
  address debtor;
  address creditor;
  ufixed originalAmount;
  currency currency;
  uint dueDate

  function loop(address loopAddress);
  function settle(address loopAddress)


}

Contract Loop {

  struct Settlement {
    Address Position;
    Ufixed anount;
  }

  Settlement[] settlements;
}
```

- Based on ethereum smart contracts
- Types of smart contracts
    - BK Smart contract: used to register invoice between BKs
        - Show the interface of the smart contract
        - Explain how we identify uniquely a company in the ledger
    - LB Smart contract: used to break a loop
        - Show the interface of the smart contract
        - Explain the process:
            - LP crawls the ledger looking for loop
            - Whenever a loop is found he creates a new smart contract summarizing the loop and offering the members to break the loop for a certain amount of DEBTs
            - Every member of the loop is notified

- ⅔ of the loop members have to verify the that the loop is real. In case the loop is broken (the LB cheated), his stake is confiscated and redistributed among the loop members
- Once the loop has been verified BK can accept or not to be part of the loop breaking, by paying the DEBTs requested by the LB. The total amount requested by the LB is divided proportinally among all BKs in the loop, based on the settled amount.
- Once the loop break has been confirmed the LB smart contract is closed and DEBTs are transferred from the BKs to the LB

2 - Core Implementation Obstacles

- Lack of an efficient, sustainable mechanism for the unambiguous identification of companies across jurisdictions. Possible solutions: support global standards such as LEI (https://www.gleif.org/en) and in the meantime come up with a “proprietary” solution, such as a database where country-specific IDs are reconciled.
- Existence of multiple competing standards for the representation of documents in the “order to pay” cycle. Possible solutions: using XBRL GL as a bridge between different standards.

3 - Solution Details

4 - Business Benefits and Future Growth

5 - Token Commercialization

6 - Team Members

7 - Token issuance