Proof of Participation (PoP): Asynchronous Byzantine Activity-Oriented Protocol

Egger Mielberg

egger.mielberg@gmail.com

17.05.2018

Abstract. We present an innovative approach to the design of a decentralized asynchronous protocol that will, *first*, let business of any kind realize any project that can be formalized on a “step-by-step” basis, *second*, allow its users to completely solve Byzantine problem with one or many contracts-related associative network (subnetwork), *third*, allow business to design, if needed, a strictly economy-based model that can be stable for a long period of time. The innovative approach is totally based on the technology of Smart Transaction [1].
The protocol is mainly focused on a practical realization of economical business contracts of any kind.

1. Introduction

At any asynchronous decentralized system there is a main task for any active process. The main task for a single process is to know the current state of other active related processes. In conventional asynchronous protocol, there is no other method besides sending a state request to all the processes of a network. In our context, the asynchronous protocol is a protocol that works with processes each of which is associated with one or several business contracts. By Smart Transactions the main task can be easily solved by dynamic verification of process’s input channel.

PoP protocol is an associative decentralized network that consists of contract-associated processes with contract-based responsibilities. The state of a process of the network is “active”, if and only if the process is a part of one or more business contracts. Otherwise, the state of the process is considered as an “inactive” one.

The message system of PoP is realized by a multicast paradigm. The multicast processes are determined by a contract.

In PoP, each process has a personal marker, "*contract's hash value*". By the contract's hash value, anyone in PoP can identify other process (participant) of the network. This principle allows PoP to be as much secure as possible. Moreover, it let a business activity in PoP be transparent and clear for its participants.

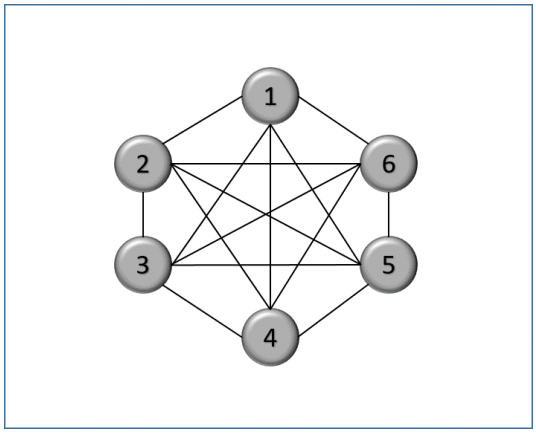
2. The main challenges of a decentralized system

2.1. Asynchrony

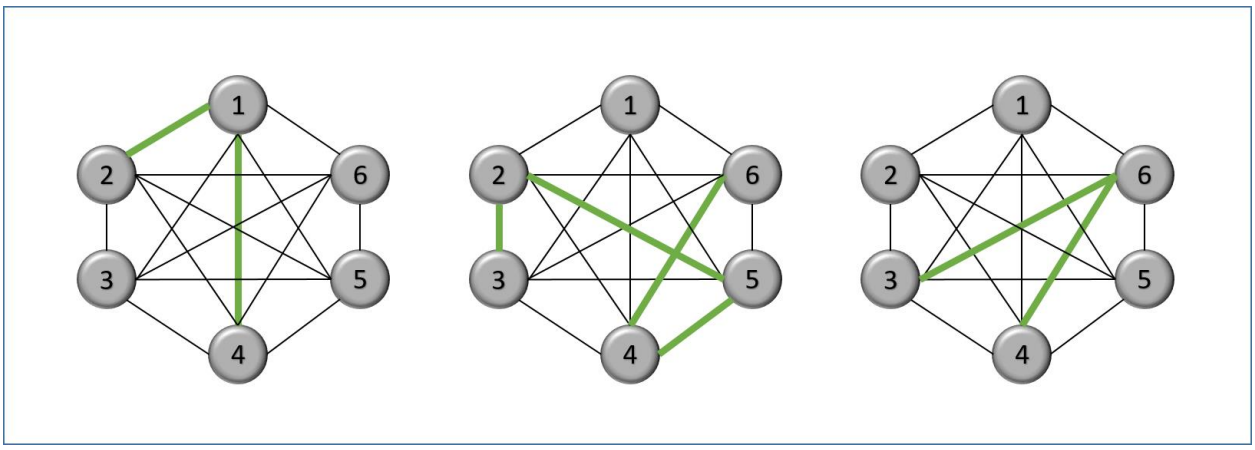
In compared with synchronous system, an asynchronous one doesn't have a centralized clock unit. In [3], the asynchronous system is considered as a sequence of rounds. In each such round, every process sends messages to all others, waits for only n - t messages of that round and changes state. The process cannot wait for more than n – t messages in a round since there is a possibility that all t faulty processes don't send any message in that round. In context of PoP there is no need to wait at all.

In PoP, every single message is a multicast one. The multicast direction is strictly determined by a contract ('s). A single process (node) of PoP network can only have one of two possible states. "**waiting**" or "**executed**". Every single contract is divided into one or several executive contract-related phases. Each phase determines a list of its active processes. Let us have a look at the following example.

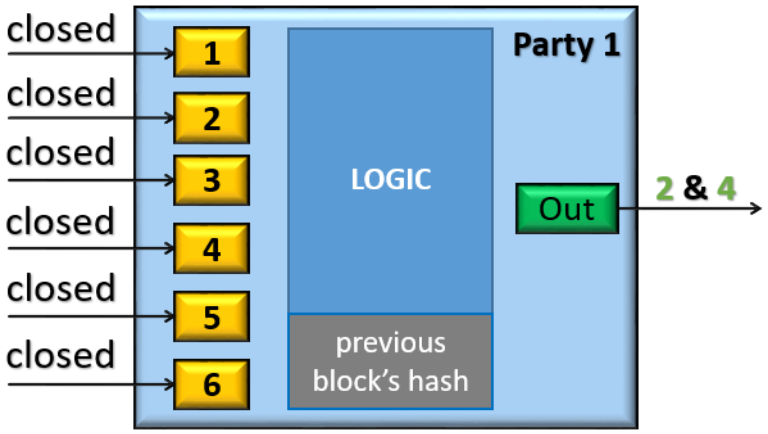
Suppose, there is a business contract with three executive phases.



Suppose, also, that the contract has six parties. The execution of the contract is divided into three steps (phases).

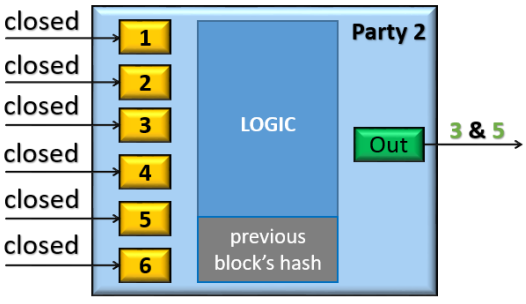


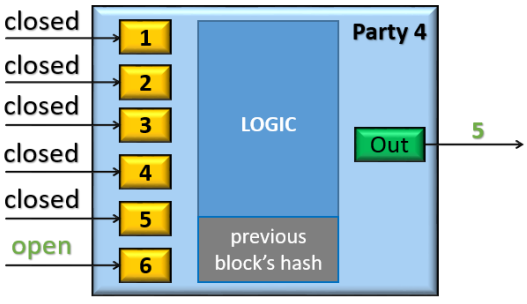
According to the contract, in *phase 1*, party 1 must execute some actions (money or data transmission, specific job execution, etc.) for parties 2 and 4. In this phase, party 1 has two output channels for party of Smart Transactions we have the following party 1 profile:



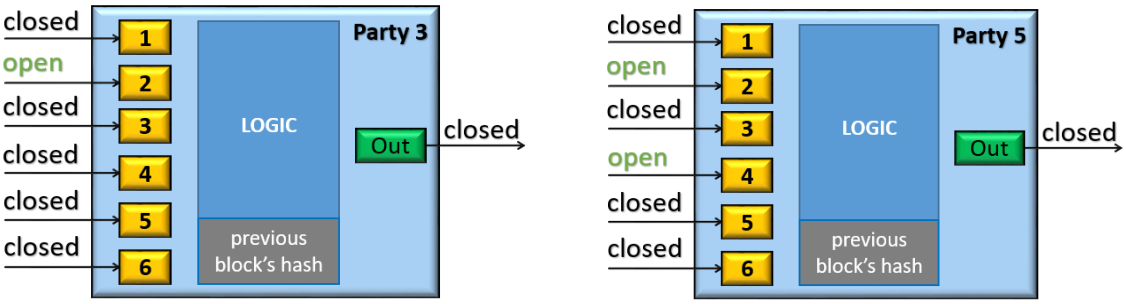
Phase 1.

In *phase 2*, party 2 must execute some actions for parties 3 and 5. Meantime, party 4 must execute some actions for party 5. The same logic is used for *phase 3*.





Phase 2.



Phase 3.

By knowing structure of each executive phase, a party can easily and promptly check the state of any party by a verification of its input channels at any time. Additionally, the output channels can also be checked in some business cases. Thus, in PoP there is no need to broadcast a lot of messages and then wait for a period of time. In other words, asynchronous mechanism in PoP is totally traceable.

2.2. Byzantine Generals problem

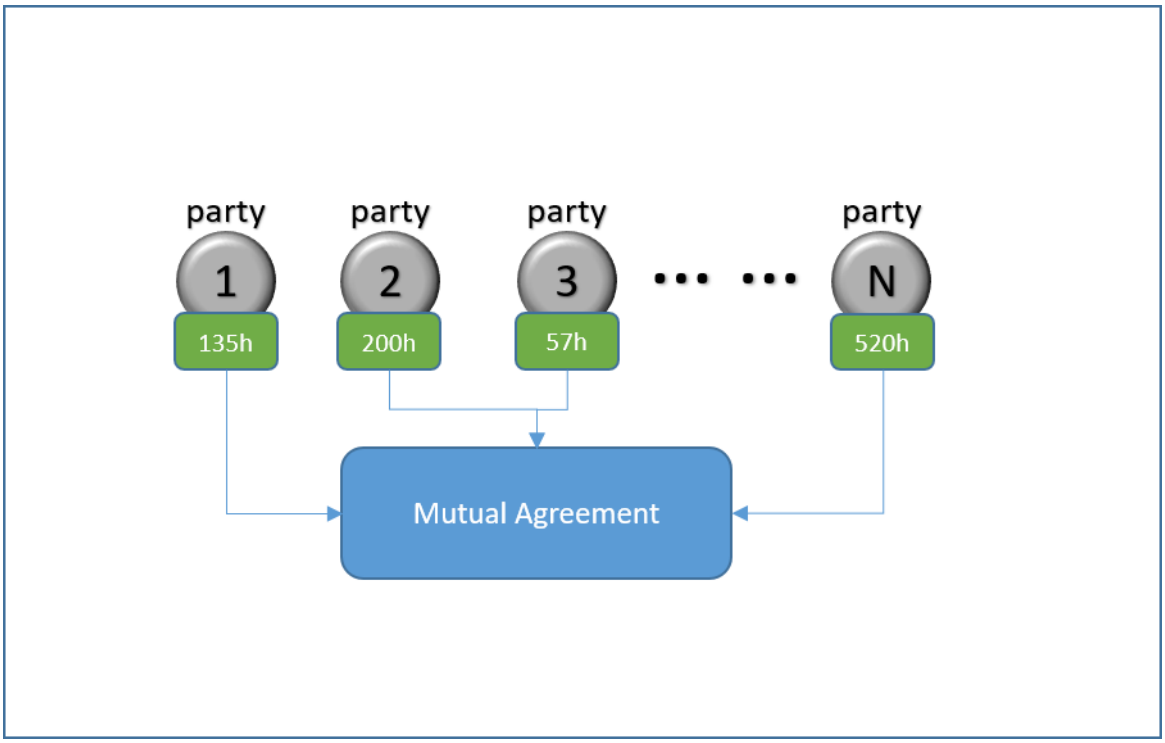
Byzantine Generals problem is one of the main and complex problems for any decentralized system. In context of computer communication system, a General can be considered as a personal computer or web server. The problem arises when one of the personal computers or servers sends two or more different messages to different recipients in case of existence of only one correct message.

In [3], authors present two solutions for the problem. First one is that if and only if more than two-thirds of the Generals (personal computers or servers) are loyal. Second one is that if any communication between the Generals is implemented by a unforgeable signed message system.

The first solution has at least two difficulties in practical realization. First, it requires development of a complex system that would be capable of identifying a loyal General. Second, it doesn't guarantee that a message between Generals will not be intercepted and forged.

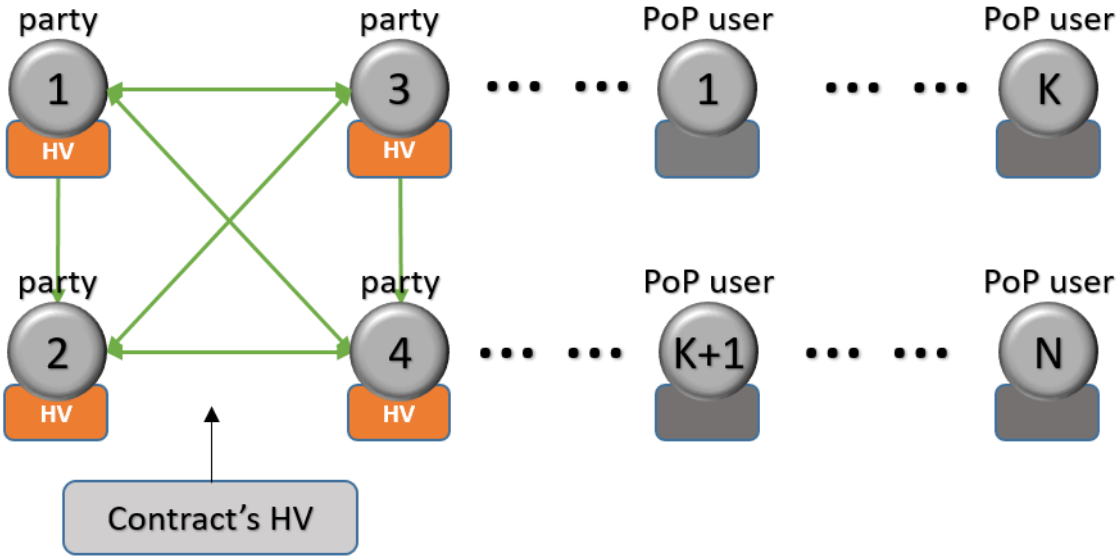
The second solution has one apparent difficulty. It requires at least two-thirds of Generals to send a message to each other for getting a consensus.

PoP-based solution doesn't have above-mentioned difficulties in their nature.



Loyalty of the contract's parties (Generals) is determined by a mutual agreement between all the parties that are engaged in a contract ('s). Each party has its own number of hourly activity. The hourly activity is calculated by previously executed contract ('s).

As soon as a contract is signed any message between the parties of the contract is secured by two hashes, "*party's hash value*" and "*contract's hash value*", respectively.



As for a consensus problem, each contract is divided into one or more executive phases. In each such phase all the parties have its own contract-related responsibilities. There is no need to vote as all the contract's activity is predetermined. During the execution of one of the contract's phase any party can dynamically check the state of other party by checking input/output channels.

In [3], authors describe a new replication algorithm that is able to tolerate Byzantine faults. However, the presented algorithm exponentially degrade its performance as soon as the number of replicas (nodes) is being increased. It is one of the crucial aspect for designing a decentralized system as increasing of complexity of a task leads to increasing of number of the nodes involved in a solution of this task.

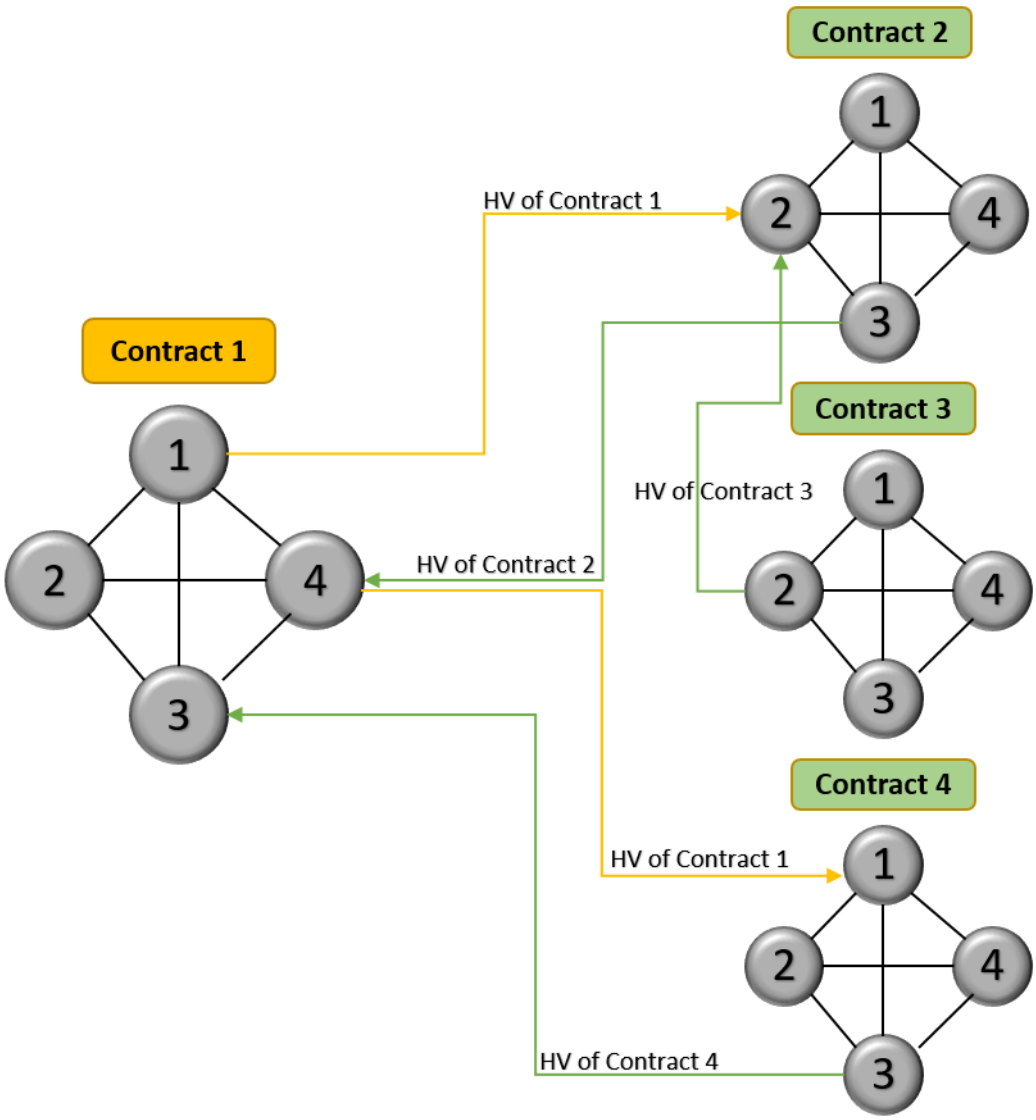
2.3. Broadcasting

In asynchronous broadcast protocol a message is simultaneously sent to all nodes in a decentralized system. However, there are many situations especially in a process of execution of a business contract where no need for some parties to be notified about other party's action. In this case, broadcast messaging can be a kind of messy.

In [4], authors present an approach which is based on a broadcast message with an acknowledgment of previously received message from other process. One of the main problems in such approach is littering a network with the acknowledgment-related information that in many cases doesn't have its targeted recipient. In other words, once a message is

received by processes of the network only one or several of them will need to know about the message's acknowledgment part.

In PoP network, multicast is used. Each multicast message is strictly associated with a contract's execution plan.



Any node of PoP network which is associated with, say, Contract 1 can work to one or many other nodes associated with Contract 2, 3, and 4, in parallel.

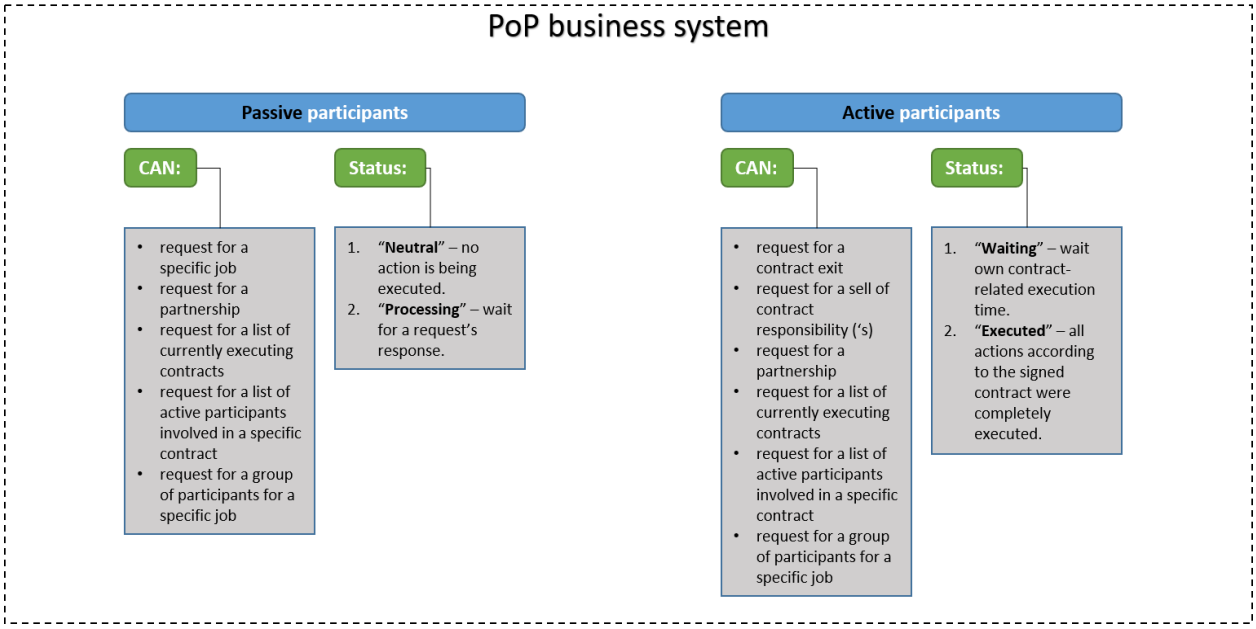
Multicast messaging is realized by the principle of "ONE ACTION ONE ROUND" (**OAOR**). *A node is being received a message if and only if it is arranged in a contract*. Balance algorithm of input/output channels for a single node is tuned by the logic of Smart Transaction Box [1].

2.4. System state

The state of a decentralized business system is crucial for two reasons, minimum.

First reason is a necessity for business to know, how many nodes are "**active**" (associated with an execution of the contract) and how many nodes are "**passive**" (no current association with any contract).

Second reason is a possibility for business to correctly calculate a total number of execution hours for a single node (contract's party).



In PoP-based business system, possibilities as for the passive as for the active participants (nodes) can be added if it is required by an economical reason. According to a newly-added possibility (function) the type of a status for both kinds of participants can be changed as well.

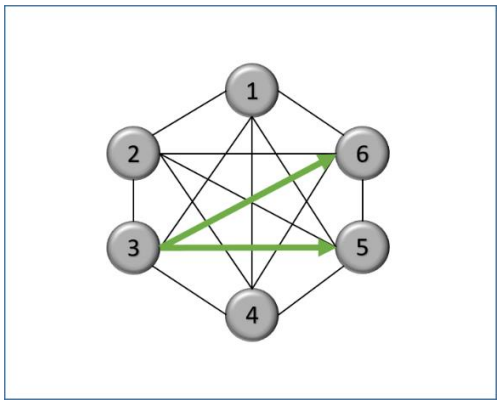
3. Consensus problem

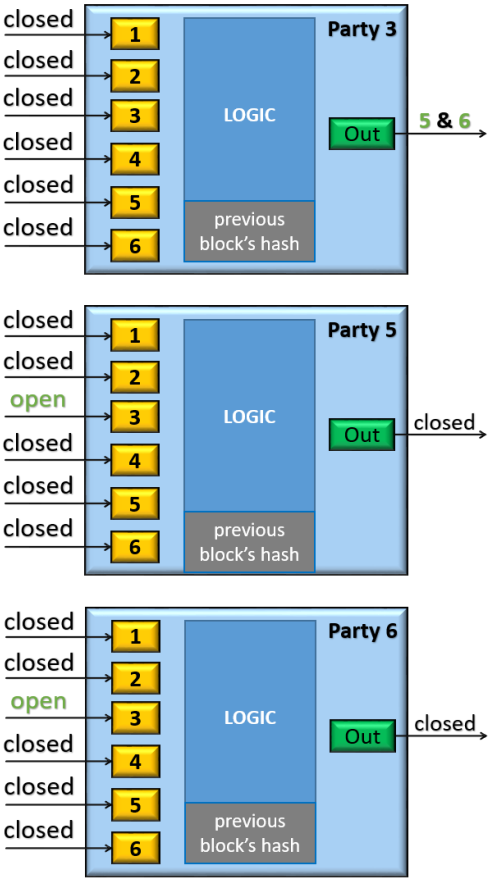
The problem of reaching agreement among a set of remote computers (nodes) is one of the most fundamental problems in any distributed system.

In context of a transaction system the problem is called as a "transaction commit problem".

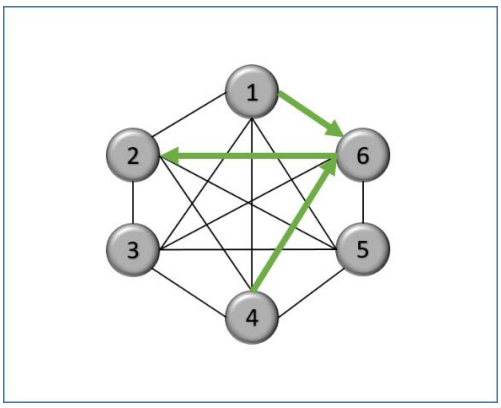
In [5], authors claim that every partially correct protocol for the consensus problem has some admissible run that is not a deciding run. They

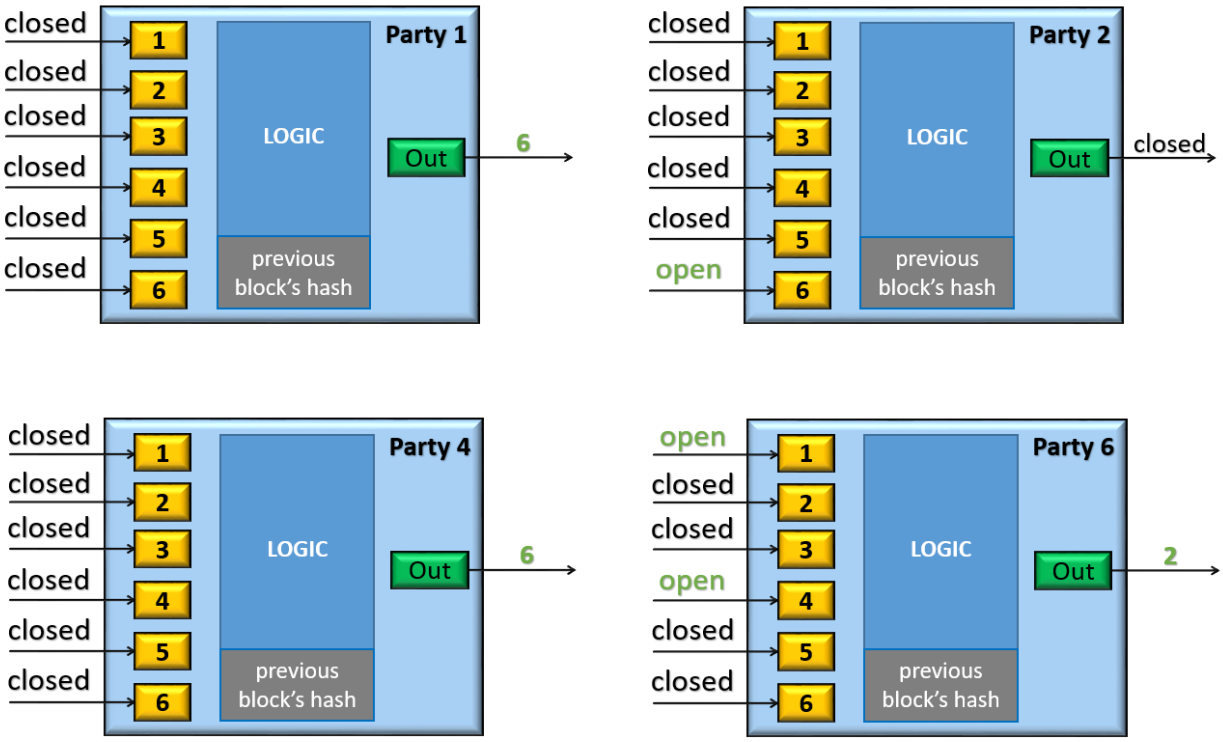
determine run as an associated sequence of steps that take one configuration of the system to another. A configuration of the system consists of the internal state of each process (node), together with the contents of the message buffer. However, we came up with another practical result. We are now confident of **two things**. *First one* is that any theoretical assumption about a faulty possible process should be considered in the context of practical contract-related collaboration between the nodes. *Second one* is that a practical consensus problem takes place if and only if there are no any predetermined relationship conditions between any distributed nodes. It becomes more clear and obvious if we use “Smart Transactions” technology.





Phase 1.





Phase 2.

For instance, six globally distributed nodes that are solely tied to each other by a signed contract can check its state through a single state request at any time. As seen above, there is only one node (3) in phase 1 with a possibility to be faulty. In that phase, only three nodes (3,5,6) are active and responsible for an execution of the contract. All the other nodes are inactive (in “waiting” state). If a transaction from the node 3 to the node 6 is not executed during the contract-determined time, the node 6 can immediately send a request for retransmission to the node 3. No other nodes are involved in such an agreement-related process.

4. PoP principles

Proof of Participation Protocol is based on **three principles** that are coming from its name. Asynchrony, Byzantine-tolerance and Economy-based collaboration.

4.1. Asynchrony

Like in a neural space, all the input transactions (neuro mediators) connect to any contract-opened “input channels” of a node (postsynaptic terminal of other neuron). The input channels of the node can be tuned on one or many different contracts simultaneously.

In context of Smart Transactions, each single node is capable of receiving thousands or even millions input transactions, in parallel. The input transactions are treated by means of associative logic (**AL**), not FIFO method. AL let PoP Protocol work steadily in an asynchronous mode.

4.2. Byzantine tolerance

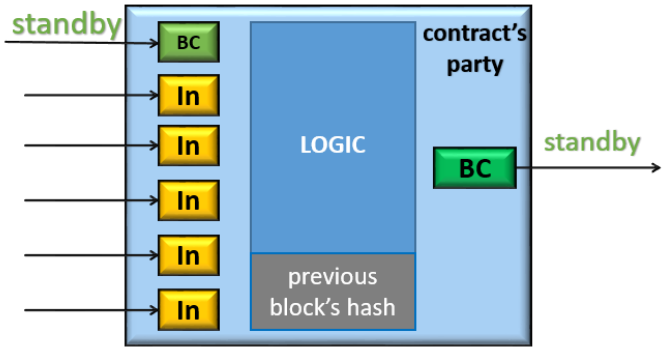
In case of failure to send a contract-related message (“execute a transaction”) the faulty node can be treated in two ways:

1. Other node (‘s) with the same protocol state sends a retransmission request to the faulty node.
2. Other node (‘s) with the same protocol state multicast a recovery request to all the other contract-related nodes by using “**Byzantine channel**”.

In case of contradictory (spurious) messaging the faulty node can be treated in two ways:

1. Other node (‘s) with the same protocol state sends a retransmission request to the faulty node.
2. Other node (‘s) with the same protocol state multicast a violation request to all the other contract-related nodes by using “**Byzantine channel**”.

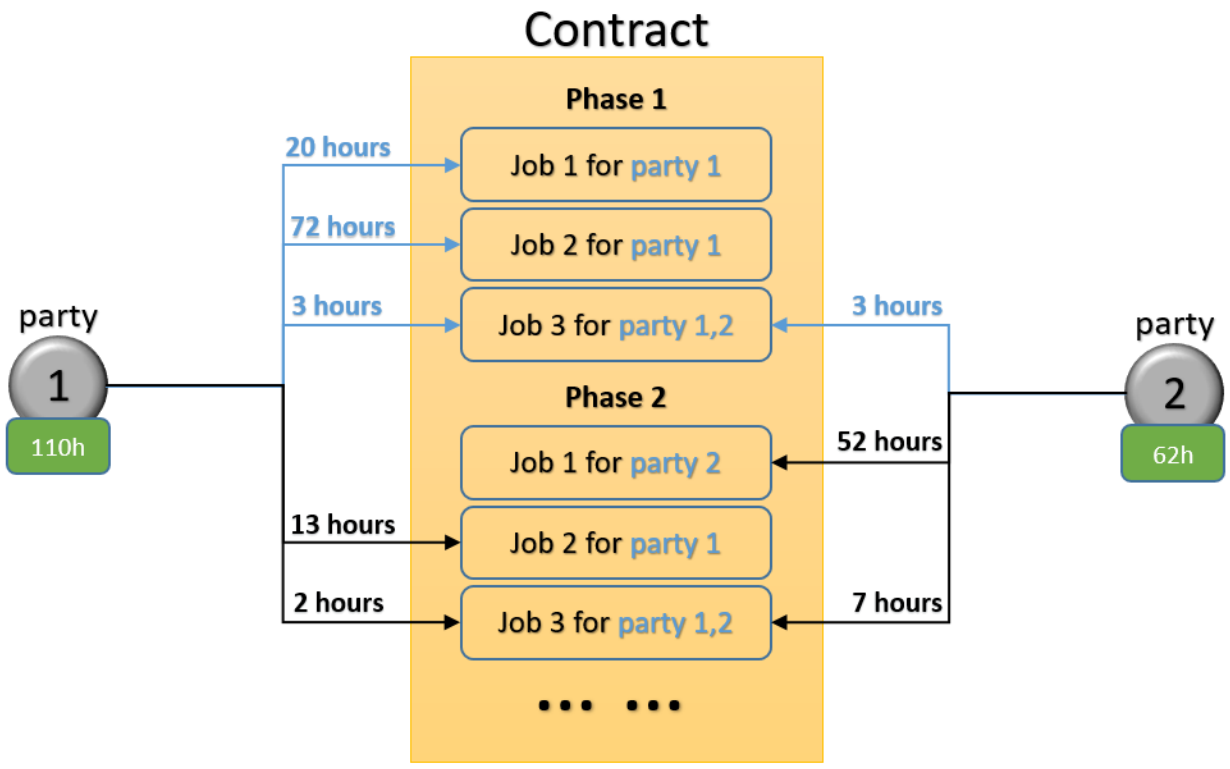
The Byzantine channel is open as soon as a contract is open and closed as soon as the contract is totally executed and closed by all parties. During the execution of a contract, the Byzantine channel is always on a standby mode. The channel is also used for a voting system.



BC – "Byzantine channel"

4.3. Economy-based collaboration

In PoP Protocol, any collaboration between two or more participants are strictly based on a signed contract. Business activity of any kind is regulated by the norms and rules of the contract.



According to the signed contract each party is assigned a fix amount of hours for each job in each phase that it has to perform. After finishing the contract each party gets a total amount of earned hours associated with the

contract. Then, at the discretion of each party the earned hours can be sent to any other participant of PoP network.

As in case of state of the decentralized system, in PoP network there are two types of the state for its participants, “active” and “passive”. In context of a single participant, “active” state means a participation in one or more contracts. Otherwise, the participant has a “passive” state.

5. Multicast messaging

One of the advantages in PoP network is that every single transaction is multicast to predetermined recipients. A recipient is always a party of one or many contracts. In most cases, during an execution of the contract the transactions are unicast between each party. It greatly clears PoP network from “noisy” messages and does it more robust and fast.

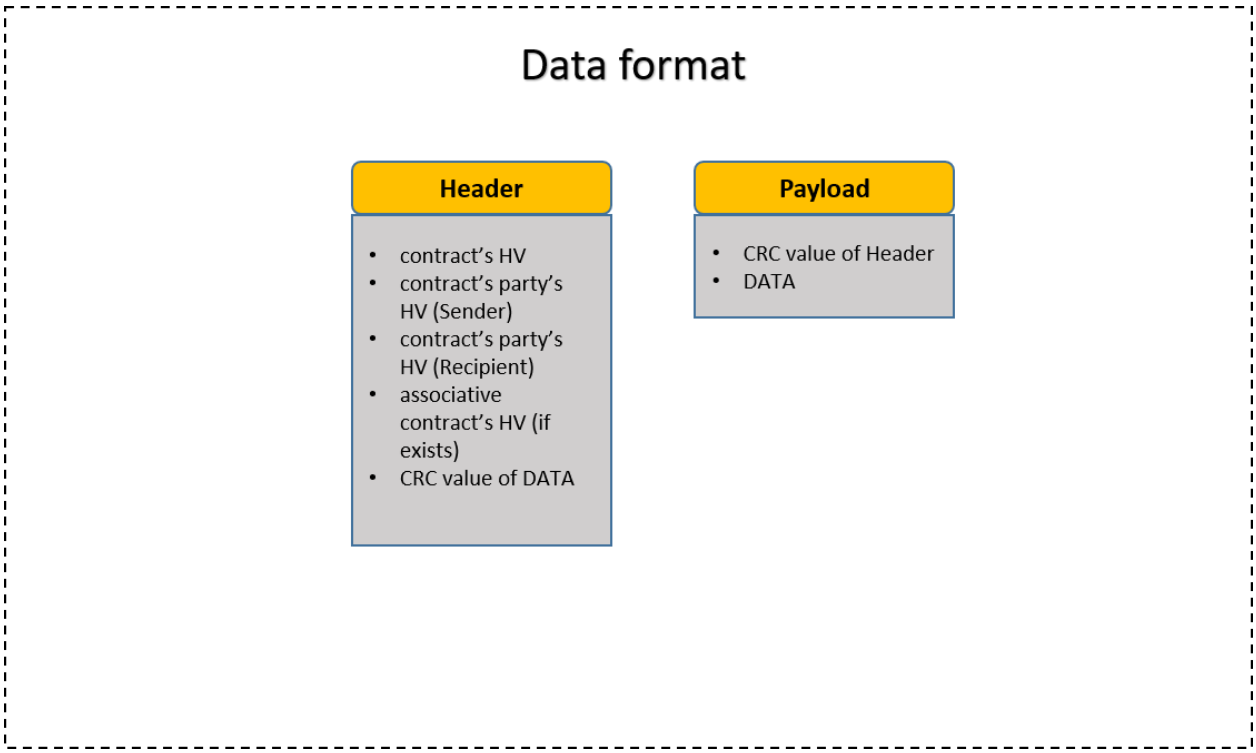
6. Protocol specification

PoP Protocol is intended to be light and quick. *Lightness* is achieved by using a direct unicast message to a predetermined contract-related recipient. In compared with TCP protocol, PoP Protocol have a series of advantages in terms of business contracts:

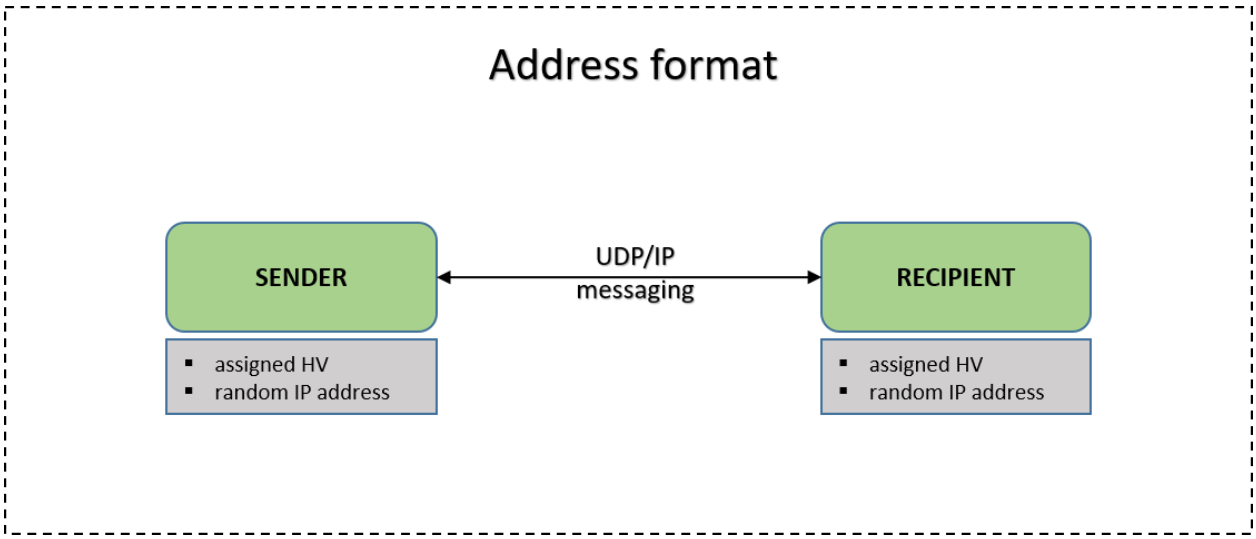
1. it *doesn’t need* any handshake procedure because right after a signing of the contract all its parties are known about each phase step of a single party.
2. it *doesn’t need* to arrange a message flow between its participants because in each contract phase there is a limited list of parties who are eligible to execute a predetermined job.
3. *there is no any necessity* to broadcast a ton of messages for each party of the contract and track it all because at any time any party can freely check any party’s state by a unicast message.

Quickness is achieved by using a UDP/IP stack along with an asynchronous contract-regulated algorithm. A single barrier for a single transaction is a party’s node’s speed. Correctness procedure for the transaction is realized on an application level.

Below is some brief specification.

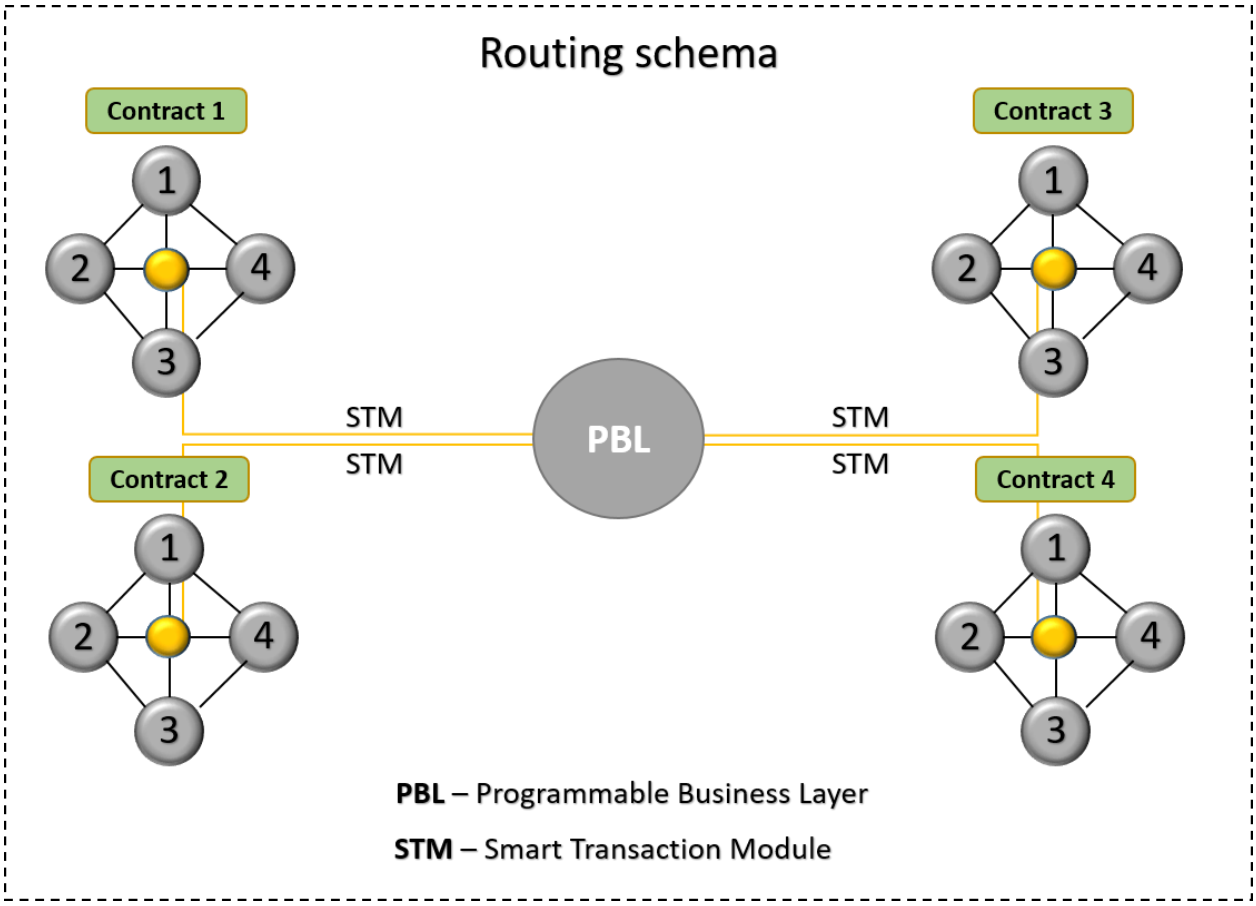


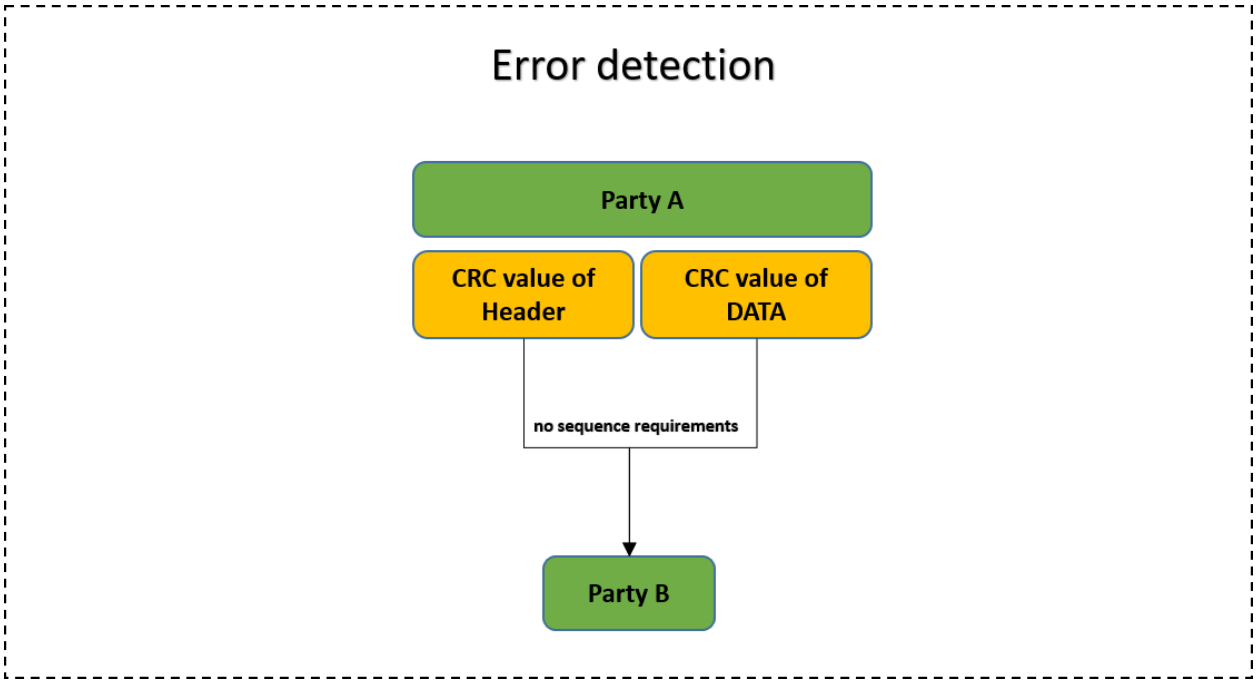
The Header and the Payload are sent by different transactions. One transaction sends the Header, another transaction sends the payload. The sequence of the transactions can be different at a given time. Such division is intended to greatly increase security in communication between participants of PoP network.



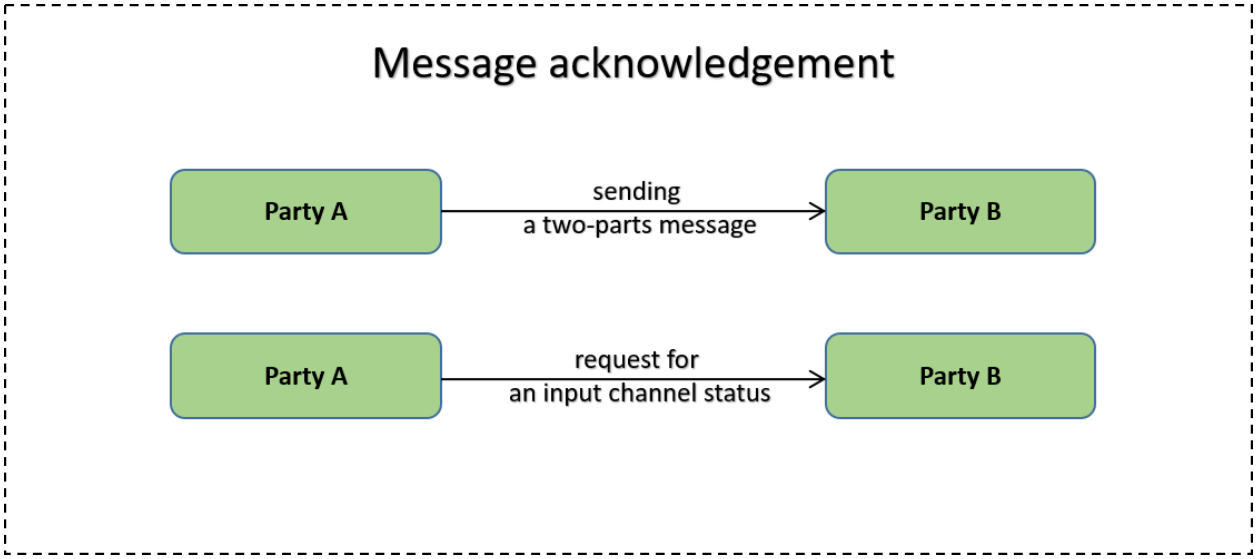
Smart Transaction Module (STM) is a module that describes the business logic for a specified contract. Each contract ('s) network can connect to each other through a generated hash value by Programmable Business Layer (PBL). PBL is mostly used in a case of business association of two or

more contract networks. As for two parties from different contract networks, one of the contract's hash value is used.

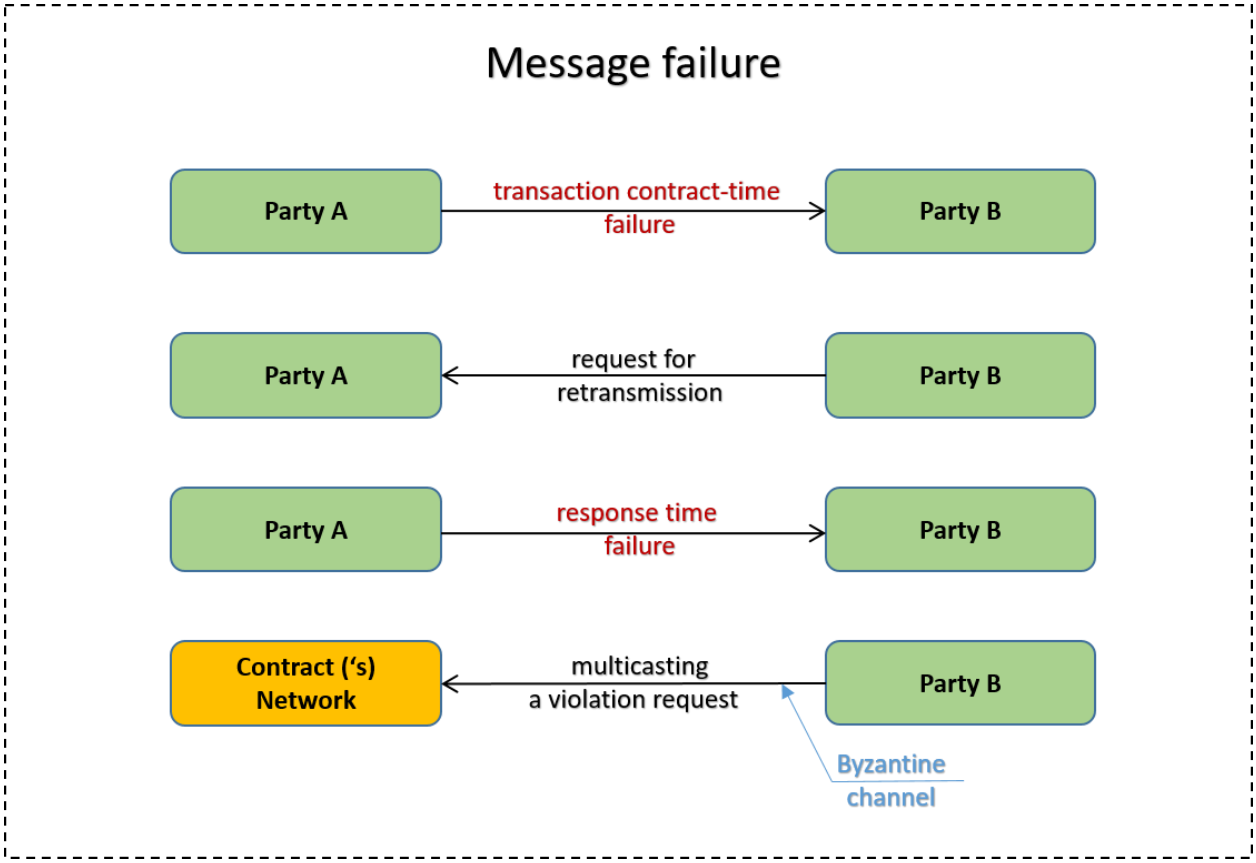




For security reason we divided a message unit into two parts. It would be harder for an attacker to hack the message as each message has a CRC value of either previous or next message unit.



As an acknowledgement of message reception, sender (Party A) can at any time check the status of recipient's input channel's status. Right after the message is received the input channel is being closed (according to a phase of the contract).

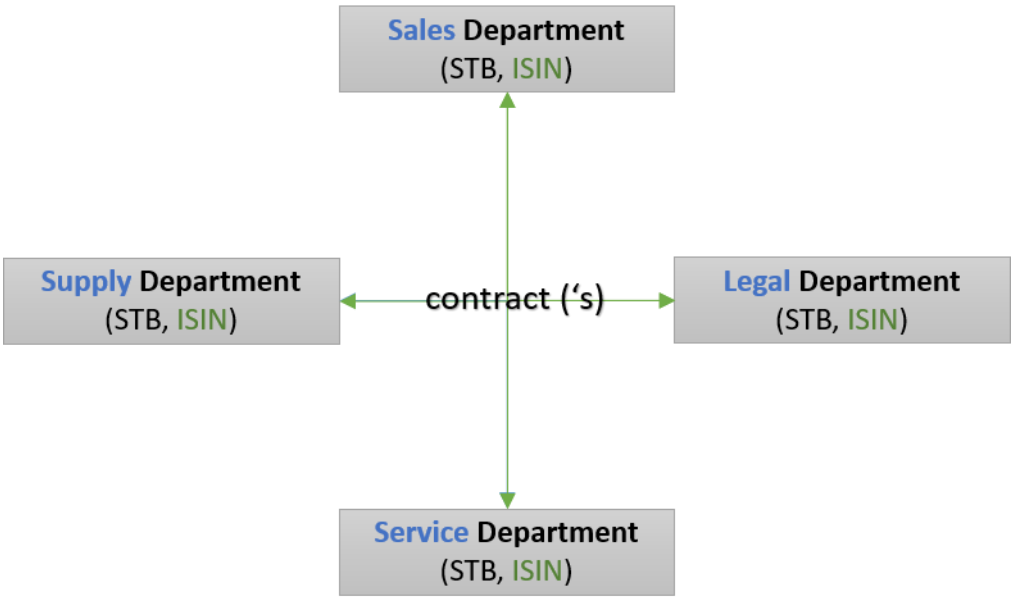


More details about specification can be found in [6].

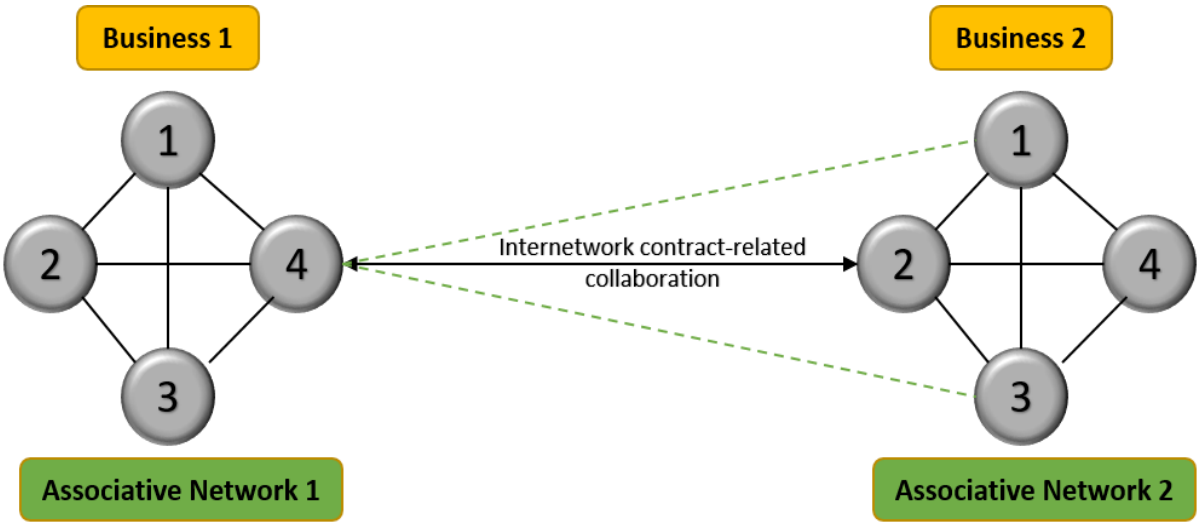
7. Protocol implementation

The main focus of the protocol implementation is a business contract-oriented network. The business network can include but not limited:

1. small business with several departments between which a service contract (‘s) can be established.



2. medium and large business with a number of various departments starting from a manufacture department and finishing with a PR department.
3. network of small business.



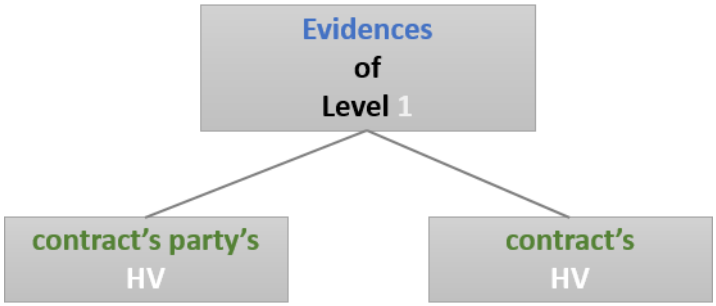
A connection between networks can be established by two or more parties of the contract (‘s).

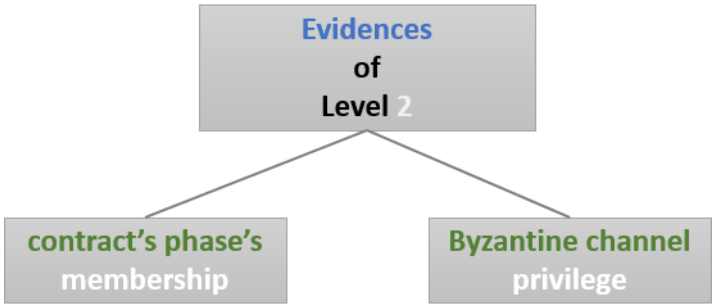
4. network of medium and large business.
5. any business cooperation between two or more participants of the network whose agreement can be formalized in a contract of any kind.

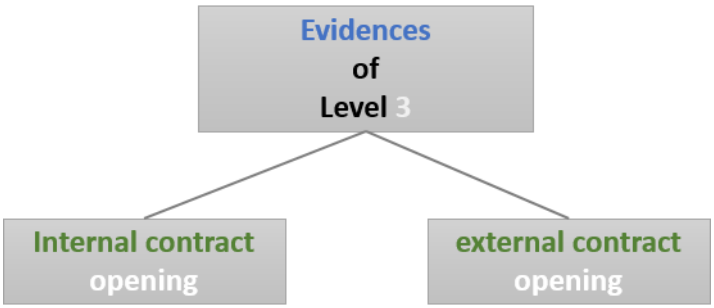
The protocol will be extremely useful in many cases where a participant of the network exactly knows what service he or she wants to get from other one (‘s). While all collaborations between participants should be contracted the protocol presents a lot of opportunities to dynamically and mutually change the contract’s conditions.

8. Validity

Validity of a transaction is a crucial component in any decentralized system. Methods and approaches to a violation of any process are different and directly depend on predetermined requirements between parties of the process. In general, the requirements are based on a list of evidences. In terms of PoP network, the structural contract-related evidences can be presented as follows:







Level 2 determines possibilities for parties of a signed contract during an execution of each its phase. It also determines whether the privilege of usage of Byzantine channel is granted or not.

Level 3 determines possibilities for parties to open a contract inside the associative network (NCN) or outside the one. Each party of a signed contract has specific requirements and possibilities in NCN.

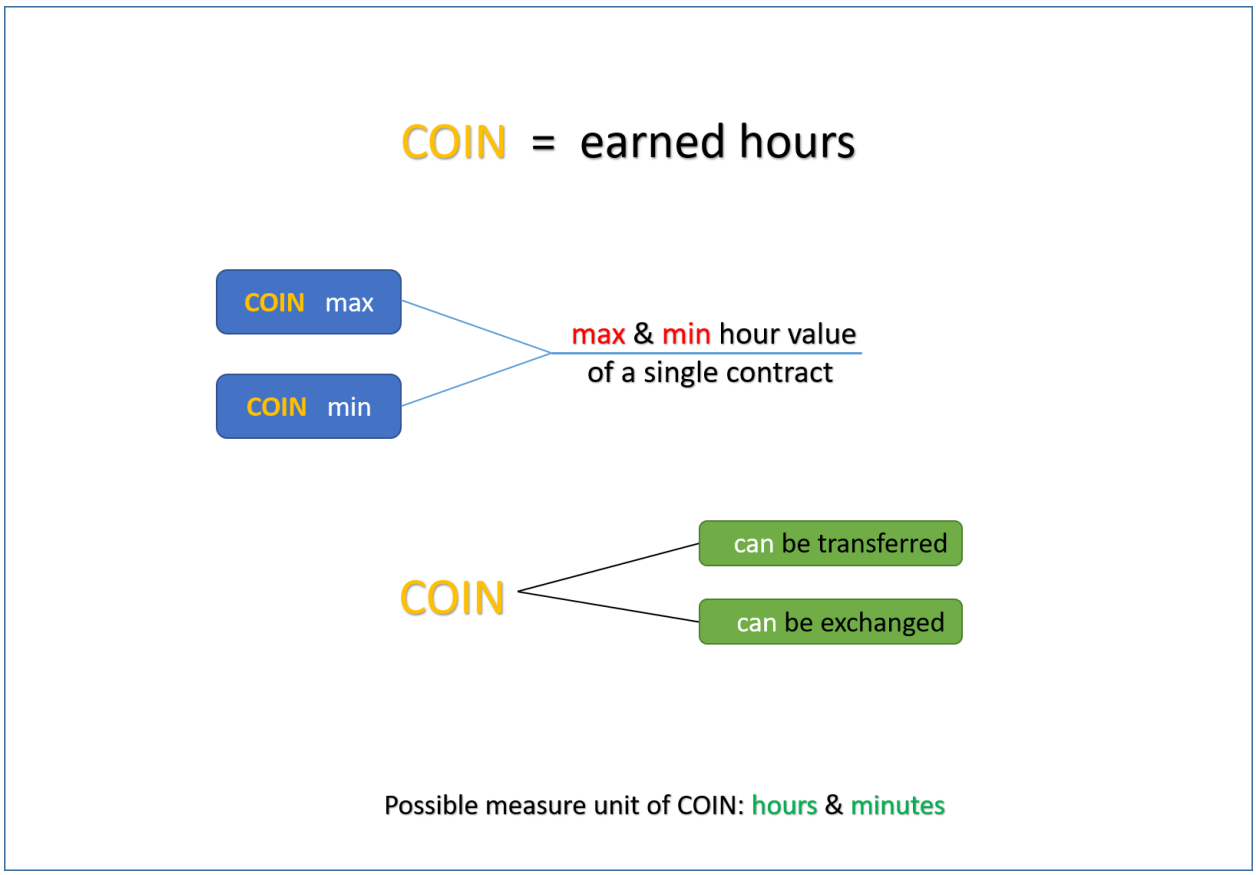
Validation process is totally realized by checking the evidences of each Level.

In some cases, the number of Levels as well as its content can be changed at the discretion of parties of a contract.

9. Stability

Independence of speculative exchange actions and other artificial market events makes any currency or stock (fiat or crypto) extremely attractive for any investors worldwide. But the task "**to get stable**" is not such a trivial one as it might be seemed. The stability is strongly tied to *economy of a system*. That is why in PoP network we are directly focusing on an economic activity of the participants. As a **measure unit** of that activity we propose *hours* (or minutes) that are calculated in a contract. Each participant of the contract ('s) is assigned a fixed amount of hours for his or her job. For example, in [7] is realized a first worldwide electronic currency that is intended to be stable by the principle of mutual beneficial activity between participants of NCN. So, in terms of finance system, the electronic currency can be economically tied to a contract's hours.

In case of a long-term investment the stability of an investment object is a *key factor*. As for any decentralized system it is a basis to start building from.



10. Conclusion

We have proposed a decentralized economical mechanism for practical realization of any business activity. The mechanism allows its users to avoid a Byzantine Generals problem in many cases. Fundamental technology of the mechanism, “Smart Transactions”, let the users create thousands of associative business networks (NCN) with millions of internetwork connections (transactions). One of the key features, “economy-based activity” let the users of PoP network build a full-scale stable business application of any kind.

References

[1] E. Mielberg, "Smart Transactions: An In-To-Out Manageable

Transaction System", 2018

[2] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", https://bitcoin.org/bitcoin.pdf, 2008

[3] G. Bracha, "Asynchronous Byzantine Agreement Protocols", https://pdfs.semanticscholar.org/510e/071390c7fbd166bee9359e79d8a68a273f66.pdf , 1987

[4] P. Melliar-Smith, L. Moser, V. Agrawala, "Broadcast Protocols for Distributed Systems", https://pdfs.semanticscholar.org/2728/48ab45b55f7452d58fb86ed8a169356edcc9.pdf, 1990

[5] M. Fischer, N. Lynch, M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process ", https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf, 1985

[6] E. Mielberg, "PoP Protocol. Specification", to be published, 2018-19

[7] E. Mielberg, "SphereCoin: A Decentralized Economy-based Electronic Currency", to be published, 2018

[8] A. Goodchild, C. Herring, Z. Milosevic, "Business Contracts for B2B", https://pdfs.semanticscholar.org/77a2/94d0703b1e8cdc33fdbca26a0d2d475cb15a.pdf, 2015

[9] Z. Milosevic, D. Arnold, L. O'Connor, "Inter-enterprise Contract Architecture for Open Distributed Systems: Security Requirements", WET ICE'96 Workshop on Enterprise Security, Stanford, USA, https://ieeexplore.ieee.org/document/555065/, 1996

[10] T. Sandholm, "Negotiation among self-interested computationally limited agents", PhD Thesis, University of Massachusets, Amherst, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.5760,1996

[11] I. Rahwan, "Interest-based Negotiation in Multi-Agent Systems", http://web.mit.edu/~irahwan/www/docs/thesis.pdf, 2004

[12] S. Kraus, "Automated Negotiation and Decision Making in Multiagent Environments", http://u.cs.biu.ac.il/~sarit/data/articles/acai01.pdf, 2001