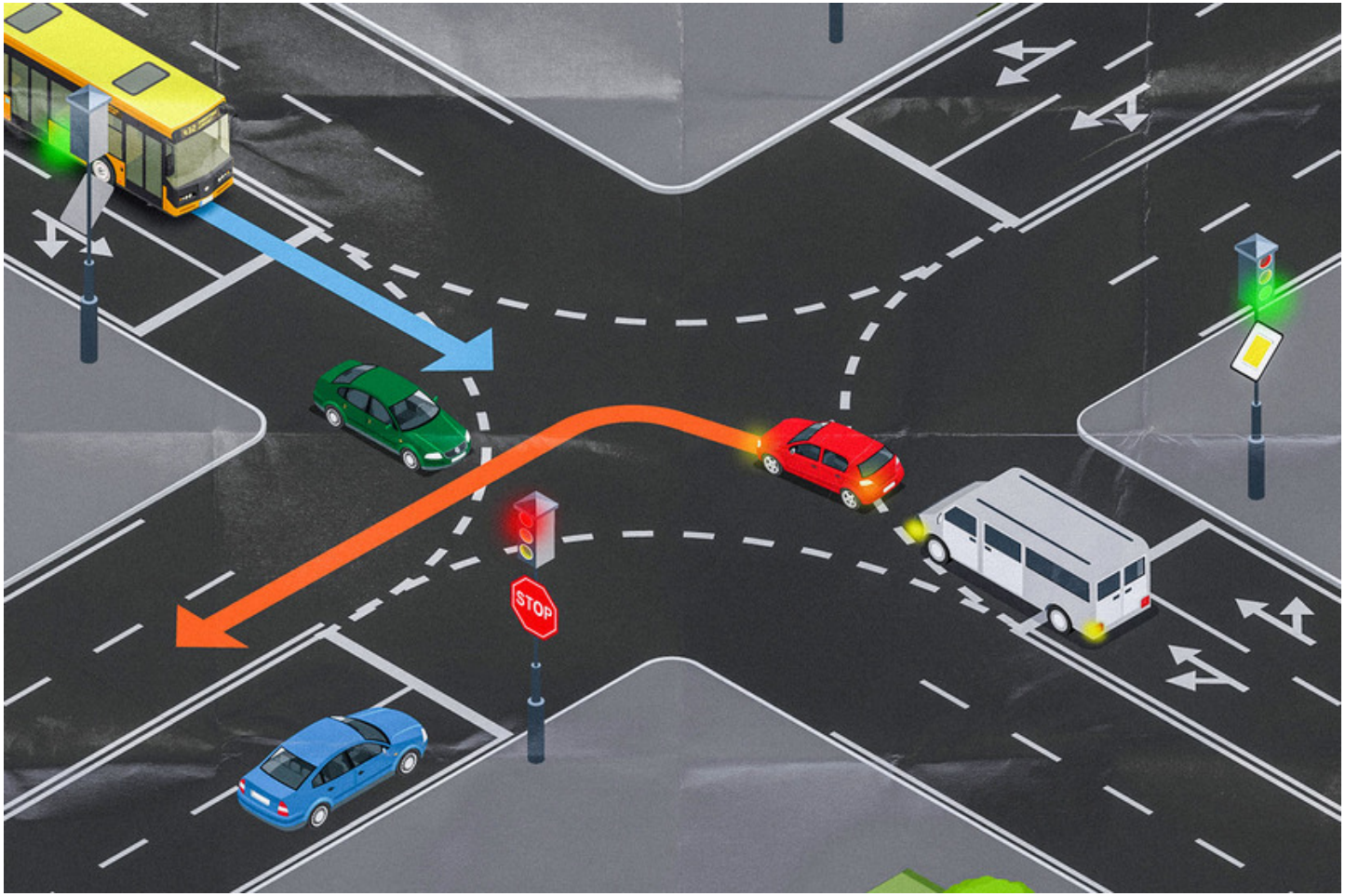
Formal methods for analyzing BlockChain

Zoran Ognjanović, Angelina Ilić Stepić, Aleksandar Perović

OpenDay, 16 may 2022

Outline


STOP

Motivational example

- it is necessary that every driver knows the rules, e.g. the driver knows he needs to stop when the light is red.
- But, in order for driving to be valid it is necessary:
  - other drivers need to know that
  - that every driver knows that others know
  - that others know that this driver knows
  - that the driver knows that others know that he knows ...
- driving rules must be generally known.
- So, what matters here is: the knowledge of the individual and the knowledge of the group **common knowledge**.

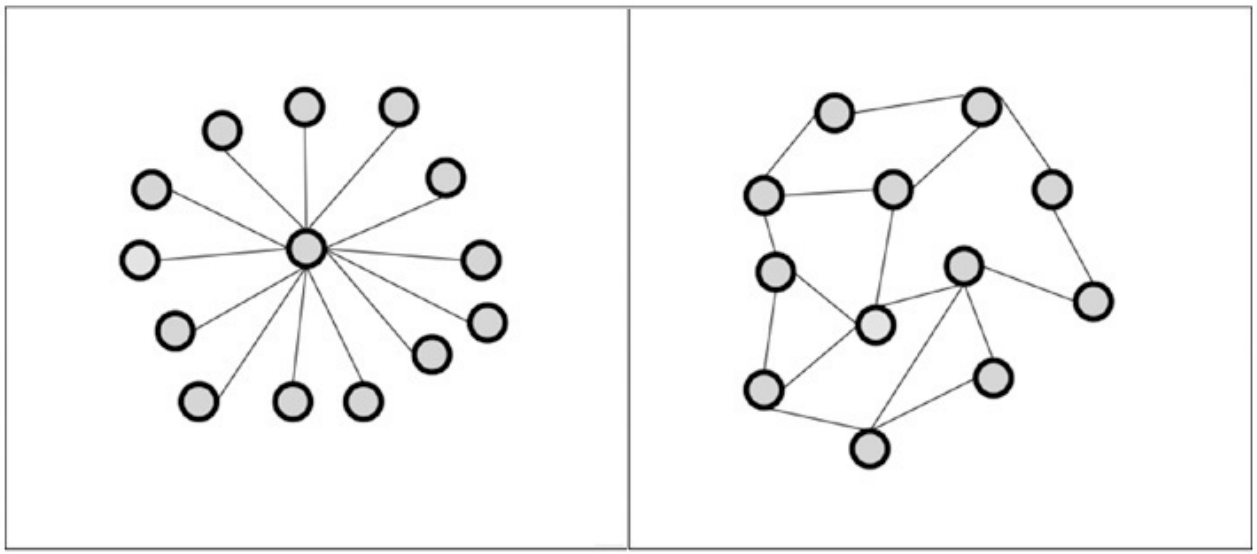
- A distributed system is a collection of (independent) components (processor /node / agent) that are located in a space (physical / virtual) and that communicate in order to achieve some common goal.
- Rational reasoning about distributed systems can be complicated.
- The source of this complexity is the inherent uncertainty in these systems, especially due to (non) delivery of messages, possible errors or unexpected behavior :processor / node / agent,
- and the basic problem is the lack of **global knowledge**, ie. knowledge of the system when the agent (does not) know what is happening in the rest of the system.

- Protocols, a set of rules by which agents behave, need to be designed **and proven** to work properly regardless of message loss and failure processor.
- The approach often used in describing and analyzing distributed systems and protocols is based on Kripke's models with possible worlds in which the agent knows φ in a world if φ is true in all the worlds the agent considers possible.

Centralized and decentralized (banking) systems

There are two types of organization (software) systems:

- centralized, with one point to which others are connected
- distributed in which components communicate without centralized coordination and / or control.



This can be applied e.g. for a money transfer system:

- banks - a centralized system, guarantee the correctness of the transaction
- BlockChain - an example of a decentralized system, in which there is no bank; we wonder how it can work.

For example, one of the identified problems, if there is no centralized control: The Double Spending Problem:

- there is no central account, so each agent keeps information in the locale
- it takes some time to transfer information online
- therefore, not all agents are informed of the changes at the same time
- A sells house to B
- information about this is sent to other agents
- A quickly sells the same house to C where the information hash not yet arrived
- B and C cannot both be homeowners.

Blockchain solves such problems by making available to everyone a complete history of transactions in the form of a public ledger.
It may happen that:

- agents can cancel (do not work), ie. work for a while and then do not work for a while
- agents can cheat
- messages do not have to be delivered.

The BC Protocol has the following features:

- manages a public chain of blocks that contains transactions
- in that chain the order of the blocks is immutable
- a new block is added at the end of the chain
- previously added blocks cannot be deleted, nor can their order be changed.

As a result, participants need to agree on the data in the chain.

Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.

Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.
- The agent collects new transactions in a block.

Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.
- The agent collects new transactions in a block.
- The agent solves a difficult task (PoW, proof-of-work) for that block.

Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.
- The agent collects new transactions in a block.
- The agent solves a difficult task (PoW, proof-of-work) for that block.
- If the agent solves the problem, it sends a block to all agents.

Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.
- The agent collects new transactions in a block.
- The agent solves a difficult task (PoW, proof-of-work) for that block.
- If the agent solves the problem, it sends a block to all agents.
- Agents accept the block if the solution to the problem is correct and all transactions are valid.

Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.
- The agent collects new transactions in a block.
- The agent solves a difficult task (PoW, proof-of-work) for that block.
- If the agent solves the problem, it sends a block to all agents.
- Agents accept the block if the solution to the problem is correct and all transactions are valid.
- Agents show block acceptance by working to create a new block using the hesh value of that accepted block.

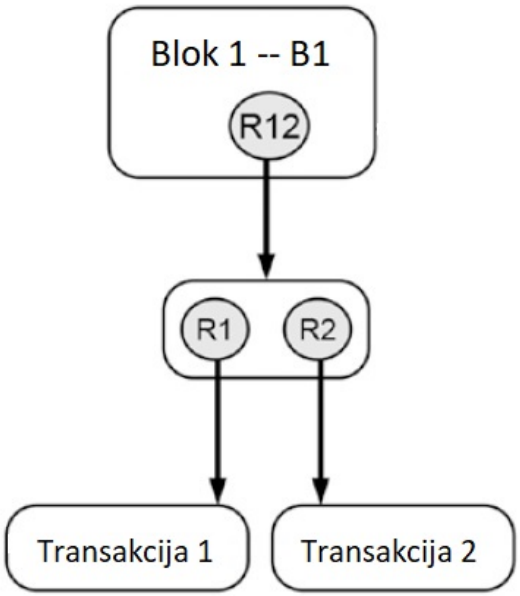
Formation of a (distributed) chain of blocks (1)

The basis for BC is given in the paper
Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. `https://bitcoin.org/bitcoin.pdf`:

- New transactions are sent to all agents.
- The agent collects new transactions in a block.
- The agent solves a difficult task (PoW, proof-of-work) for that block.
- If the agent solves the problem, it sends a block to all agents.
- Agents accept the block if the solution to the problem is correct and all transactions are valid.
- Agents show block acceptance by working to create a new block using the hesh value of that accepted block.
- Agents are always considering the longest chain and trying to extend it.

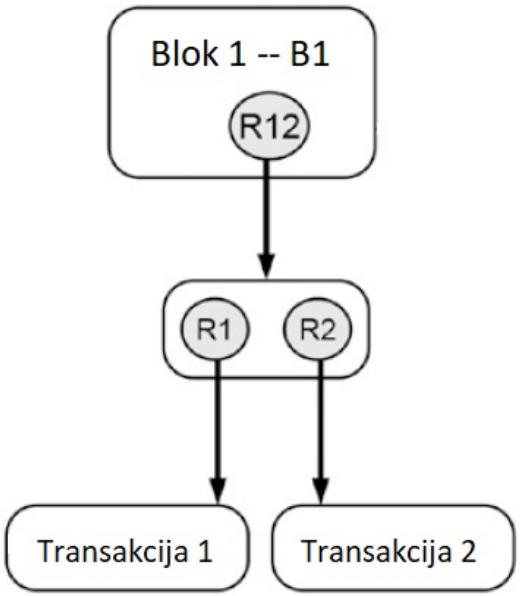
Formation of a (distributed) chain of blocks (2)

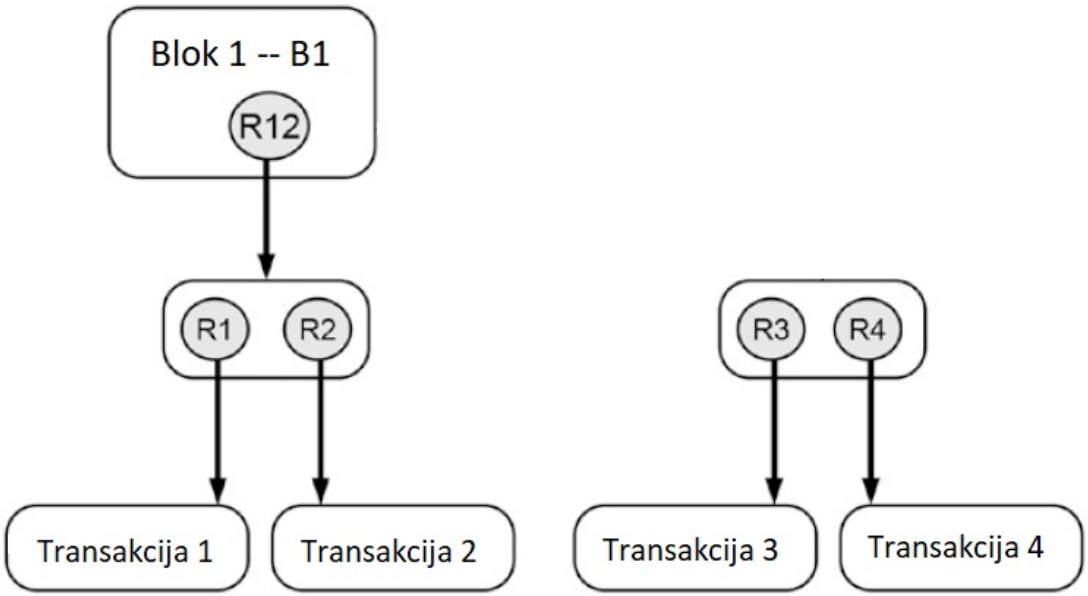
The above steps constitute one ore implementation of the Nakamoto Protocol. As it looks from the point of view of one agent: The agent maintains his local copy of the chain, called BLOCKCHAIN. Let the current chain be composed of one block as in the picture.

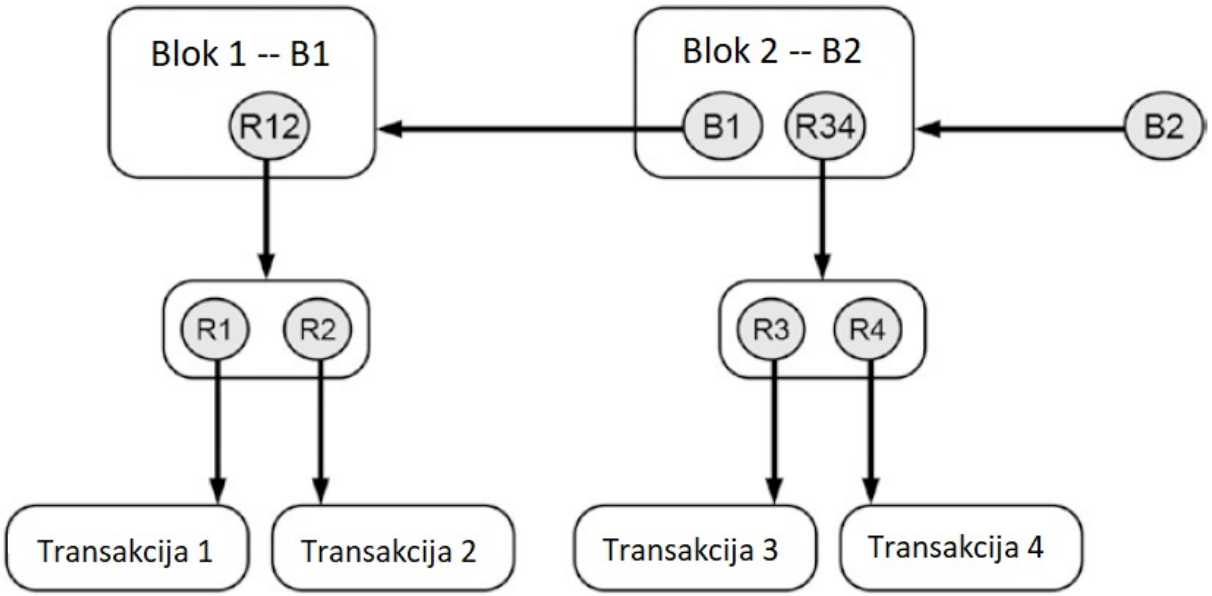


The block consists of data - transactions and their hesh values B1. In one round, the agent collects new transactions and calculates a new hesh B2 for a data set that contains:

- those transactions
- B1, as well as
- solution of a cryptographic problem (“ cryptographic puzzle ”), i.e. a number such that the resulting hesh value starts with a certain number 0.


Blok 1 -- B1
R12
R1
R2
Transakcija 1
Transakcija 2


Blok 1 -- B1
R12
R1
R2
R3
R4
Transakcija 1
Transakcija 2
Transakcija 3
Transakcija 4



Difficult task:

- when calculating the hash value, find a number such that the hash value starts with a fixed number of digits 0.

If this problem is solved the agent adds the block to the chain and sends it to the other nodes and onwards there is a B2 link to continue the chain. This procedure is called **proof-of-work** and has the property that it is difficult to do, but the solution is easy to check, ie. agents who receive this block can easily check if the solution is correct. It should be said that other procedures are possible to extend the chain.

Here you can immediately see **how to protect data**: if someone tries to change a transaction or an entire block in a chain, it changes the hash somewhere in the chain, which then changes the next hesh values in the chain, which is hard to calculate, and change is easy to detect without these costly changes because checking is cheap.
Hashes make the blocks immutable. If someone tries to change some data in the chain:

- the hash function guarantees that the hash value of that block will change and
- this would be revealed when passing through the chain.

Fork: chain splitting and the longest chain criterion

It is possible that in one round two or more agents send their blocks almost simultaneously to extend the chain. Such a situation is called FORK and they are resolved in the next rounds when new blocks arrive that will extend some of the existing branches.

The longest chain criterion:

- the agent tries to continue the chain with the first block obtained, he keeps the other blocks obtained in that round
- when one branch becomes longer than the other agents move on to it.

- For example, let blocks C and D be given for the initial chain A-B
- about half of the agents will get C or D first, then half will continue with C and half with D, but keep the other branch for later
- let the agents that continue the chain A-B-C give blocks E and F
- these are now longer chains and agents continue to work on them
- finally, the agents working on the chain A-B-C-F give block G
- it is now the longest chain and agents continue to work with it

Thus, FORK is disbanded in later rounds according to the longest chain criterion with the idea that most, or ultimately all agents, move to a single chain.

Consistency, ie. t -consistency

Establishing agreement of agents around the chain is called **protocol consistency**. More precisely, it turns out that insisting that agents agree around the whole chain is unrealistic, too strong. For example, due to message delays, it is possible that not all agents received all the blocks. The corresponding form of consistency is the so-called t -consistency:

- agents agree on all but possibly the last t blocks in the chain
- those blocks on which the agents agree are guaranteed to remain in the chain

- But even that is a very strong condition. More precisely, the protocol should guarantee a high probability of agreement on this long common chain prefix. Such consistency is called t -consistency.
- The consensus of agents is formalized in the form of COMMON KNOWLEDGE, which is a necessary and sufficient condition for simultaneous coordination.

Plan:

- in the language of logic *PTEL* expres (write) some elementary properties of the blockchain protocol (axioms) and t -consistency
- show that it follows from the axiom that agents achieve common knowledge of t -consistency.

In bitcoin, resolving forks means prevents that an attacker will win a double spend race (i.e., to succeed to spend the same money more than once) to replace the blocks in the ledger.
Consensus is especially important in the case when there is a need to guarantee simultaneous coordination of agents, e.g. to conclude a contract.

THANKS FOR YOUR ATTENTION