FAKE NEWS

A TECHNOLOGICAL APPROACH USING BLOCKCHAINS

Summary

[1 Meta-info 3](#_Toc519863177)

[1.1 Intro 3](#_Toc519863178)

[1.2 Premis Architecture 3](#_Toc519863179)

[1.3 TOVE Traceability Ontology Excerpt 4](#_Toc519863180)

[1.4 5W MODEL 5](#_Toc519863181)

[1.5 IPFS 7](#_Toc519863182)

[1.6 Validation phase 8](#_Toc519863183)

[2 ARCHITECTURE 9](#_Toc519863184)

[2.1 INTRO 9](#_Toc519863185)

[2.2 PROOF APPROACH 10](#_Toc519863186)

[2.2.1 Storage 11](#_Toc519863187)

[2.2.2 Hybrid 11](#_Toc519863188)

[2.3 FIRST BLOCK 12](#_Toc519863189)

[2.3.1 Post a Resource 12](#_Toc519863190)

[2.3.2 Evaluate 5W 13](#_Toc519863191)

[2.3.3 Consensus 5w 15](#_Toc519863192)

[2.3.4 Check validity 5w 15](#_Toc519863193)

[2.3.5 Consensus Trustiness 17](#_Toc519863194)

[2.4 SECOND BLOCK 17](#_Toc519863195)

[2.4.1 Revisited Mining 18](#_Toc519863196)

[3 Implementation 20](#_Toc519863197)

[3.1 Project Structure 21](#_Toc519863198)

[3.2 Start Point 22](#_Toc519863199)

[3.3 Cassandra Database 22](#_Toc519863200)

[3.4 Queue 22](#_Toc519863201)

[3.5 First Layer 24](#_Toc519863202)

[3.6 Second Layer 24](#_Toc519863203)

[4 Main Use Case 26](#_Toc519863204)

[5 REFERENCES 27](#_Toc519863205)

# Meta-info

## Intro

The main idea behind the blockchain, that has an important role not only in financial aspects is the used of a distributed ledger. Essentially we are going to save many blocks one after the other but, for a deeper analysis, we have to define what we want to save.

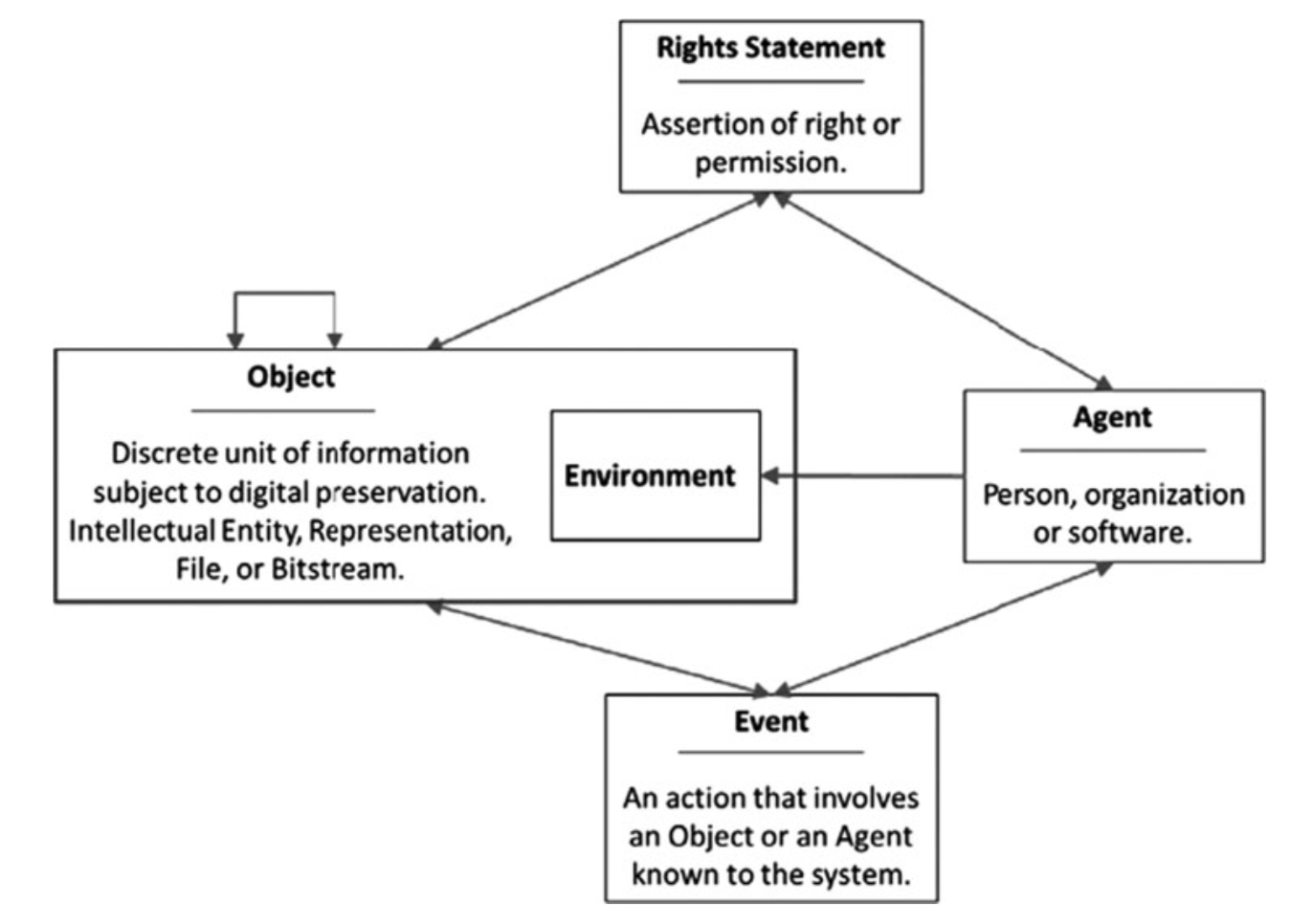
At a very basic level, metadata can be considered as statement about a resource. Digital resources are usually identified by different means, as URIs or DOIs. However, these have two problems associated:

* They rely on some trusted party or authority, as in the case of the DOI and
* Some of them (as URIs) cannot be guaranteed (and are not intended to) retrieve the same resource over time.

In a context of untrusted parties, digital resources should be identified via mechanisms that uniquely identify them from their content.

## Premis Architecture

PREMIS stands for ‘‘Preservation Metadata: Implementation Strategies’’, it outlines a provenance schema which helps identify a resource.

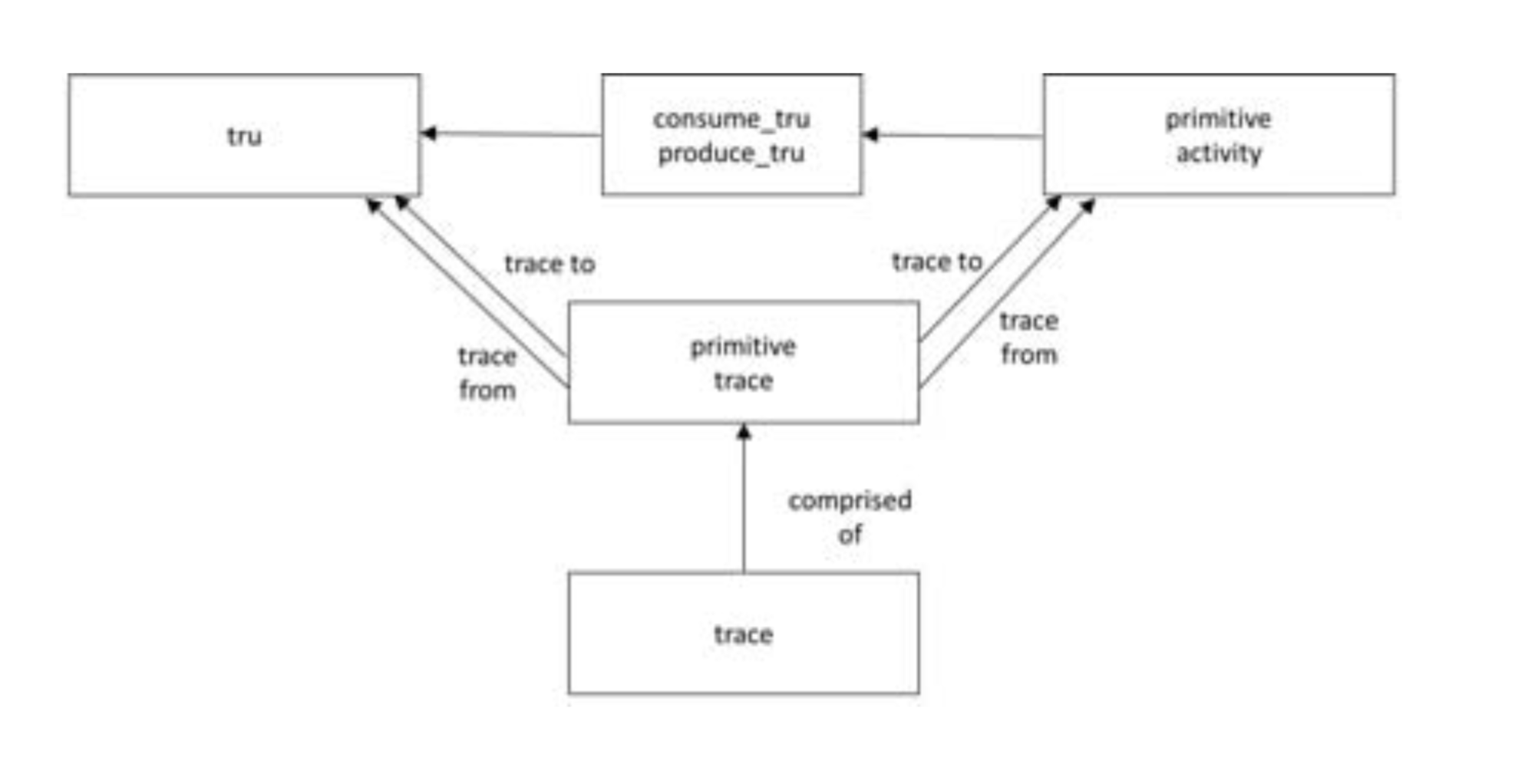


Starting from this schema we may describe what is important from our perspective point of view:

* AGENT: Not needed – if some malicious process intercept a good news to be published he may decide to sign it as first. It would be necessary an asymmetric key schema and as a matter of fact we need an authority in which we may trust. In a blockchain anonymous context we want just to add true news, every person is equal to the others, so we may skip the part concerned with the AGENT that has published the news. Those ideas will be exploited later
* EVENT: It is not useful to know (ex image capture or report written). It matters just the content published.
* RIGHTS: It is useful due to sharing of content. Ex. If we load an image we have to know if it can be edited, reposted or similar. Knowing the rights is useful in VALIDITATION part. However if we discuss a public blockchain this part does not have any importance. We will focus on it later.
* INTELLECTUAL PROPERTIES: It was used in the original paper of premise architecture. It is a good point where to start generalizing the idea above IPFS.
* OBJECT: It is the main resource. We have just to readapt the data to be stored.

## TOVE Traceability Ontology Excerpt

Editing the previous schema, it is useful an ontology based one.

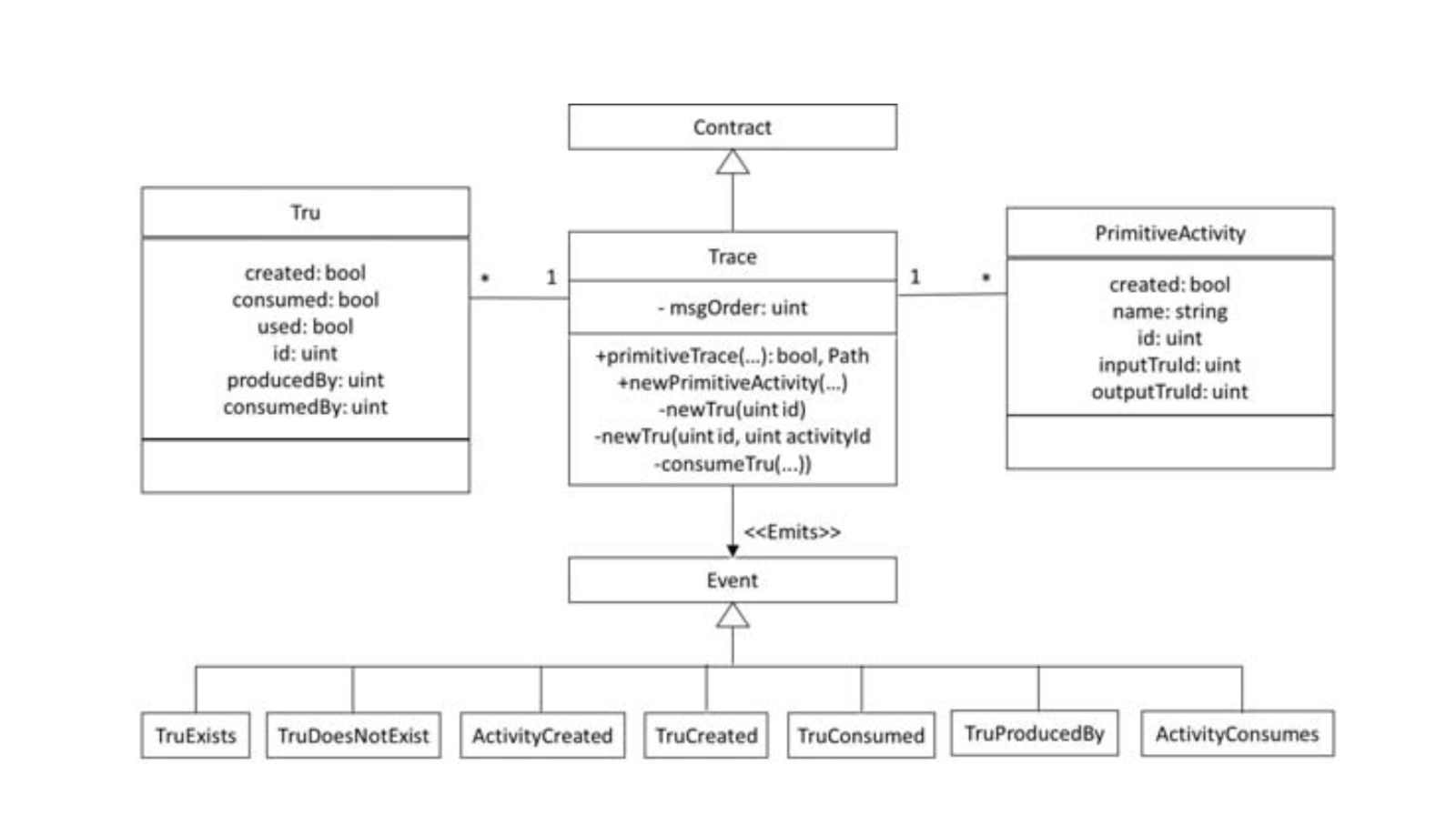


In TOVE, it must be possible to trace from one entity to another, where neither the entities are abstracted entities. TRU is the resource representation that must be traceable, since a TRU is neither an abstracted nor aggregated entity. The activity itself is a content to be exploited later (similar to EVENT in previous schema) that may even be skipped.

A modeling approach based on formal ontologies, like in TOVE, can aid in the formal specifications for automated inference and verification in the operation of a blockchain (smart contracts as “pieces of software that represent a business arrangement and execute themselves automatically under pre-determined circumstances”)

Summing up we need a TRACE (as a primitive class needed to store transaction) that correlates meta-data with the resource published, so TRU has to be defined better.

However even if we are going to redefine the schema that is behind this project, the main interesting aspect is that it provides a class diagram that is useful to write code upon smart contracts.



The only public interfaces are provided by the Trace class, therefore all user input affecting the contract state on the blockchain is through the Trace class. All output communication from the Contract is accomplished using Events, which are log data variables to the blockchain and in turn read by the Ethereum client

## 5W MODEL

This last solution is just a refinement of the previous schemas, where in particular we just describe better the composition of the main object.

This approach is well used not only to well describe the resource published (it is commonly used in journalism) but being well explicative it may be used to aggregate information that needed to be extracted in the validity phase:

WHERE

(Enviroment)

WHEN

(Date)

WHO

(Actor)

WHAT

(Activity)

WHY

(Reason – Not mandatory)

Those five main blocks (or four being one optional) well describe every resource. Every one is well explicative but 2 main things has to be pointed out:

* For each resource, there can be multiples sentences that describe it (example an article may describe different aspects)
* WHO is both SUBJECT and OBJECT (notice the arrow). Being it useful when we have to retrieve information we split sentences (maybe in replicated manner):
  + Alice met Bob in Italy
  + Bob met Alice in Italy

However, we can avoid of splitting sentences (due to avoid replication), we can use an optional WHO as DATIVE, or better the one being affected by the state or action.

Given a sample sentence such as the previous one, we may understand that a factor useful to describe the 5W is not only the main content but even the ACCURACY (saying they met in Italy or in Rome or at Via Ariosto(Rome) is completely different for extraction and so validity against other news). However the accuracy as it is presented is well difficult to describe, we may imagine

Summing up the main things said until now we may describe (in a JSON like format) the metadata:

{

    name: "example.pdf",

    hash: "0x0123abc",

    rights: {

        edit: false

        share: false

    },

    included-resources: ["hashOf1", "hashOf2"],

    content: {[

        who: {

            name: "Alice",

            accuracy: 0.7,

        },

        what: {

            name: "met Bob",

            accuracy: 0.7,

        },

        where: {

            name: "Italy",

            accuracy: 0.2,

        },

        when: {

            name: null,

            accuracy: 0,

        },

        why: null], ...

    }

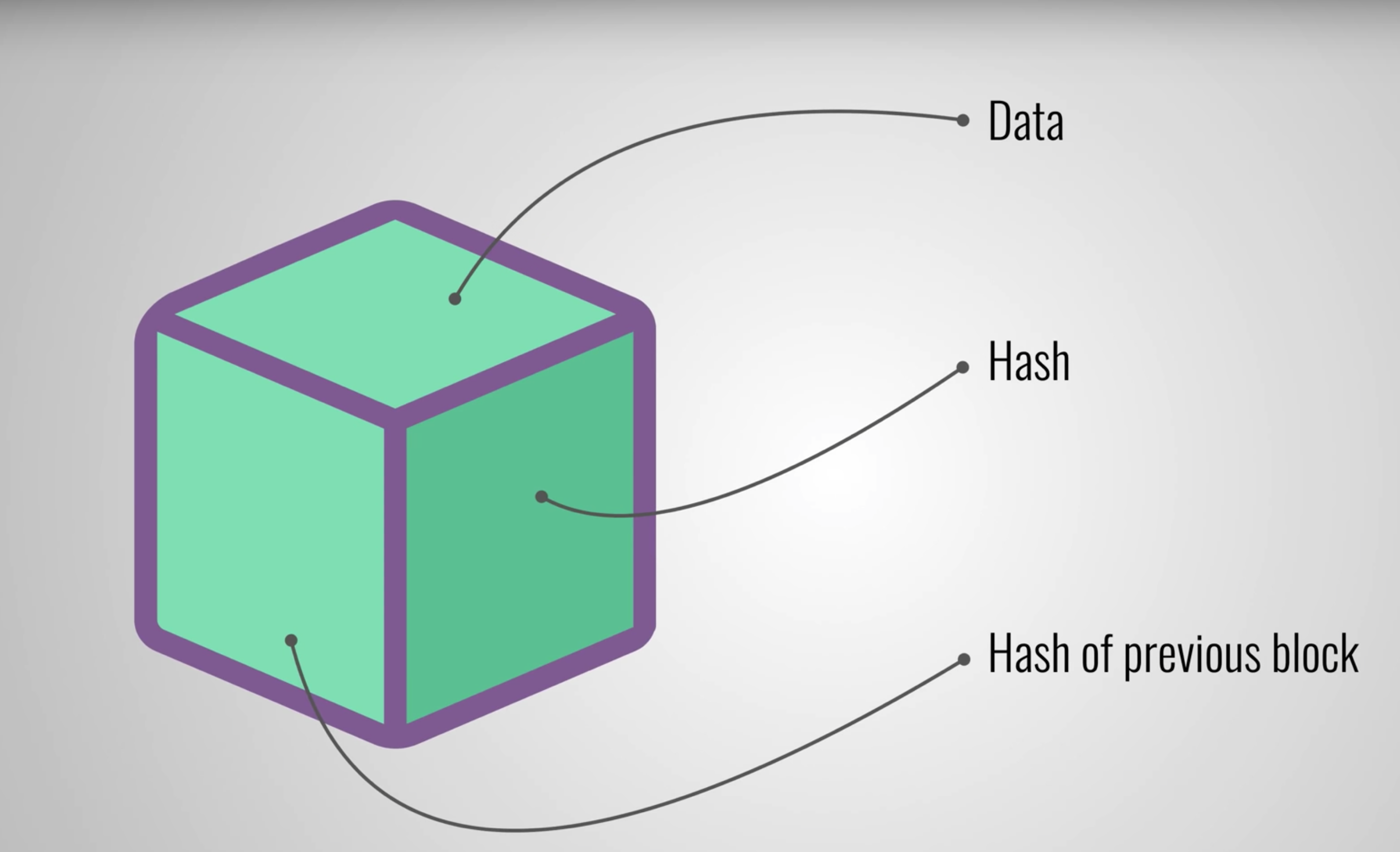
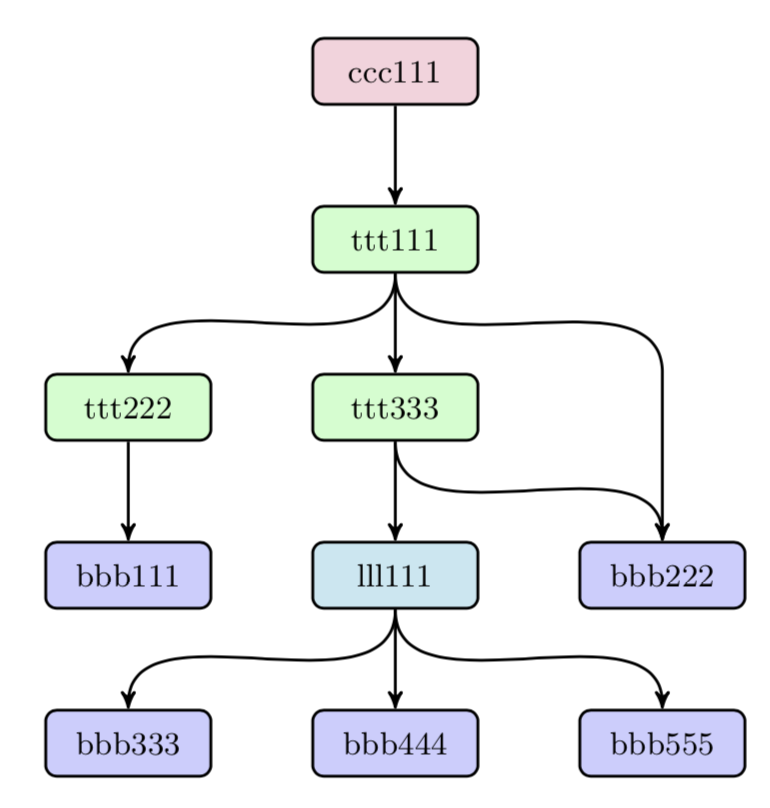
}

We have already discussed the 5w model, so now let us discuss the other parameters:

* NAME: name of the resources . file format
* HASH: cryptographic hash value of the resource, it may be further analyzed describing the type of algorithm used.
* RIGHTS: operation that can be performed on a resource, so possibility to edit content or to share it
* Included-resources: other resources linked (IPFS tree), for example an article that has some media content

## IPFS

Considering the following structure with reference to the IPFS architecture and how data in each block have to be organized we may have the following schema:



IPFS deploys a generalization of a Merkle directed acyclic graph (DAG) to establish a decentralized network of trusted data. The fundamental principle behind a Merkle DAG is that if you have the hash of the root node, and the hash came from a trusted source, then, as long as the hashes match that of the root, you can trust all leaf nodes

The main elements to point on evidence in this phase are essentially the hash and the “INCLUDED RESOURCES” tag that does not need to be declared, in fact being a resource a child of another one it is implicit the declaration of resources used due to last claimed hash.

So now, readapting the meta-information schema we have :

{

    hash: "0x0123abcd",

    claim: "0x0123defg",

    meta-info: {…}

}

Another important question to answer is “after validation phase, should we store/reject a transaction or we should always store but with a certain trustiness ?”. In principle we want to avoid fake news spreading, but it may happen that something relevant is going to be published for the first time, so we want to accept the news but be suspicious with reference to its correctness.

So another field “trustiness” has to be added with a value between 0 and 1. This may give an hint on consensus algorithm to be adopted which involves to accept or reject news according a certain threshold.

## Validation phase

Now, knowing how data is going to be stored (further meta-info can be discussed or added/removed in a more detailed analysis in practice) we have to define the main operations for signing a Transaction.

We have to notice that we will not describe the operation for checking the trustiness of a news, in practice a user that wants to check the validity of a resource essentially has to simulate the registration of a new transaction and so collect the outcome of this operation.

[UML SCHEMA ?]

The main steps are:

1. Get a cryptographic hash of the digital media resource.
   * Check whether that hash exists on the blockchain.
   * If the hash exists, retrieve the associated metadata and go step 4.
2. Create the meta-info of the digital resource.
3. Retrieve hash of media linked the uploading resource.
4. VALIDATE.
5. Sign the transaction that stores the cryptographic hash of the digital resource, and its associated metadata and claim, on the blockchain.

The first step involves the use of a cryptographic hash function (we are not going to invent, we may use one already existing ad used).

The second step involves the use of machine learning algorithm (on a media or on text) in order to extract the 5w schema.

The third step is just a search in order to retrieve resources used (a search by resource name and type for example, or we just evaluate the hash of a resource to be added and check if it is already present, otherwise we are just going to insert it with the same 5w schema found at the step before).

This step is the most important and it gives as output a numeric value between 0 and 1 and according to this value we may decide if accept or reject a news. The main operation that have to be performed are:

* Retrieve the meta-info of linked resources.
* Check the rights of those linked resources.
* Check the semantic compatibility of the 5w model of those linked resource with the one to be uploaded.

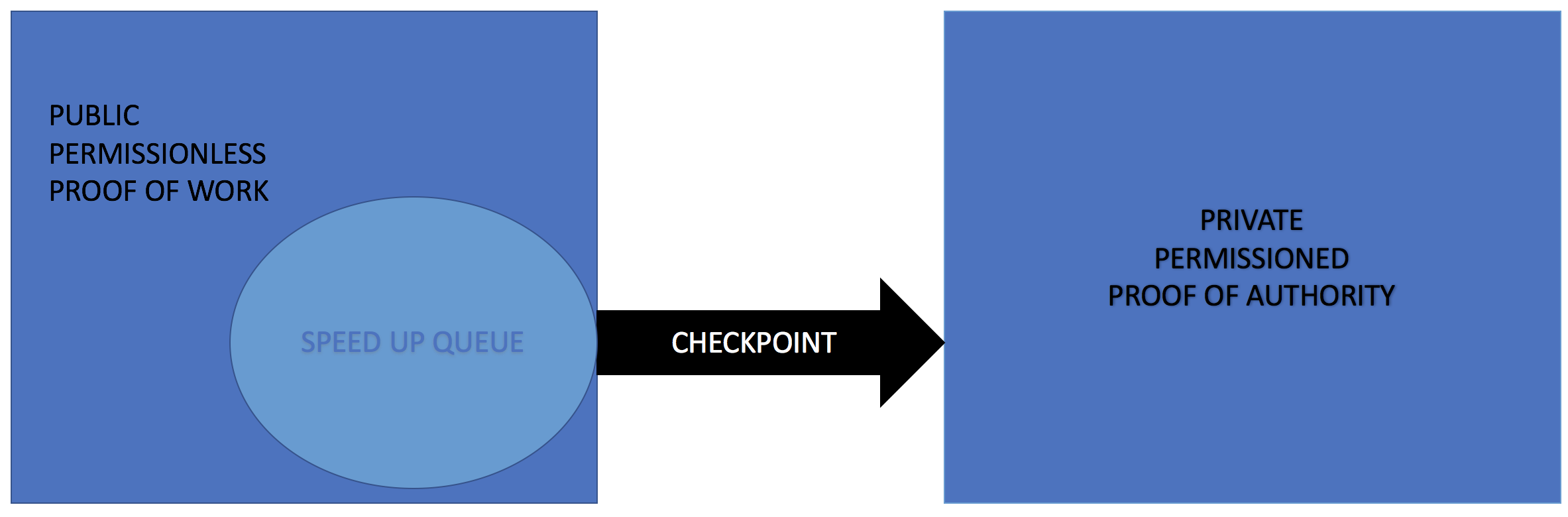
Remembering that a resource file can be described by many 5w sentences, discarding the ones whose ground of trustiness has already been checked due to the ground of trustiness of the claimed resources we have to assure the validity of the other ones (if present):

* Check the semantic compatibility between the already checked sentences and the other ones.
* Check the semantic compatibility between each other of all the sentences.
* For each sentence check the validity consulting the ledger.

The algorithm as it is non efficient, however it describes by an high point of view the main steps.

# ARCHITECTURE

## INTRO



Starting from the meta-info’s structure discussed previously we may point out our needs.

* Every news published at the end must be definitively validated
* Everyone can post and see news approved
* Only trusted entities can give a final evaluation
* Avoid spread of fake news in the heavy validation phase

The blockchain architecture essentially consists of 2 main blocks, the first one, public permissionless, is just like a filter, where we decide if we may let pass a news in the heavy validation phase; the second one represents the last layer of our blockchain architecture, private and permissioned, where we perform the second and last step for the validation of a news.

The decisions about the choice of a first public layer can be resumed in the following way:

In a public layer, everyone can post a news, otherwise we realize a kind of monopoly of information. The structure of meta-info fits better in a public approach so there is no need of using a public-key cryptography. Every news has the same initial score of another, we cannot hack an account and publish whatever we want. In addition we have to remember that presenting a 3-layer architecture, the validity of the news is evaluated better in the last layer, where we have a permissioned blockchain, we just need a ground of trustiness according to the content

Due to this first analysis, we can cut off the “rights” part on the meta-info block because it does not have any sense in a public blockchain layer, where we are just interested to the content of the news and not to the people who reported it. In fact, to give more evidence to the news one of the main content is the ability to report it anonymous so, as a matter of fact, “rights” within the schema can only be a source of danger if considered.

{

    name: "example.pdf",

    hash: "0x0123abc",

~~rights: {~~

~~edit: false~~

~~share: false~~

~~},~~

    included-resources: ["hashOf1", "hashOf2"],

    content: {[

        who: {

            name: "Alice",

            accuracy: 0.7,

        },

        what: {

            name: "met Bob",

            accuracy: 0.7,

        },

        where: {

            name: "Italy",

            accuracy: 0.2,

        },

        when: {

            name: null,

            accuracy: 0,

        },

        why: null], ...

    }

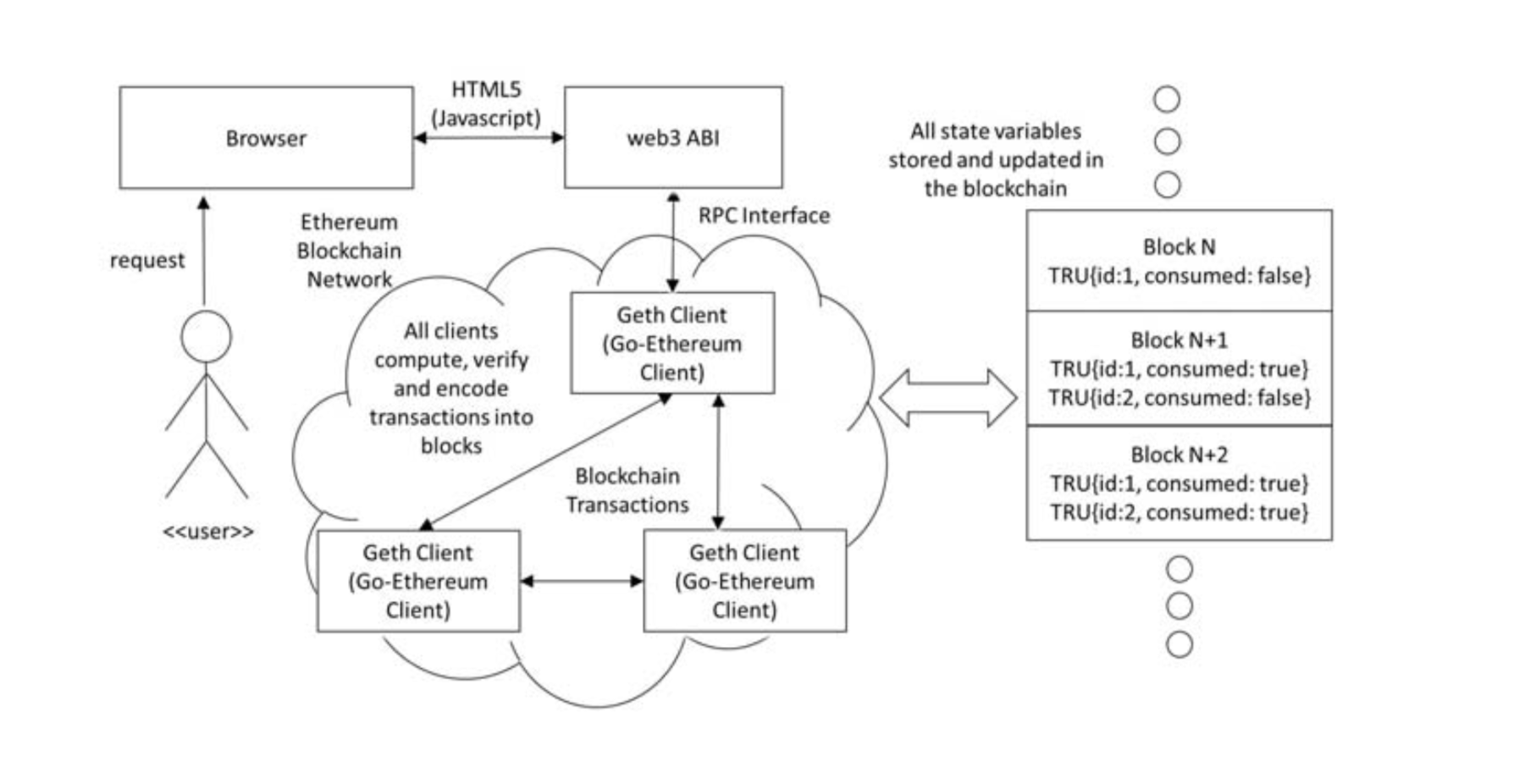
}

Summing up all we can affirm that we need a 3-layer blockchain. The first two are based upon Proof of Work, but the first layer can spread a news directly to the third one if certain conditions are satisfied (it is just a queue), the last layer instead is based upon Proof of Authority.

## PROOF APPROACH

One of the first things to discuss is the choice between “Proof of Work”, “Proof of Stake” and “Proof of Authority” concerned with our architecture design.

The first block can be realized considering the PoW approach in fact it can be realized as an implementation of the smart contracts above an already build Ethereum blockchain, as we have already seen dealing with meta-info structure for the “Provenator Architecture”.



In fact, what we need for this purpose is just a Blockchain network client (i.e. geth) is used directly or through an intermediate Web layer to deliver data onto the Blockchain.

In the context of this proof-of-concept, anonymous data is aggregated and stored using a relatively simple mechanism within a Smart Contracton the Blockchain network, implemented using the Solidity programming language. The data is combined with expert knowledge encoded within the Smart Contract to implement our meta-info schema.

Being the first layer divided in two parts, we can say that a first level is used just to speed up operations concerning with information that have already been checked. There is no need here to use a proof mechanism, it is just a queue, it can easy be implemented even in Java programming language, essentially it may be used to skip some validation’s phases and store data in a more efficient way.

The main decision in between PoS or PoA in the second block of the blockchain. The proof-of-stake (PoS) mechanism works using an algorithm that selects participants with the highest stakes as validators, assuming that the highest stakeholders are incentivized to ensure a transaction is processed, meanwhile, PoA uses identity as the sole verification of the authority to validate, meaning that there is no need to use mining.

Due to requirements that we have in this last layer, the need of a news of being correct and the use of trusted entities as peers in a private blockchain, PoA fits better than PoS in fact, this last one in this case can be seen as a generalization of its concept where the amount of stake is fixed and equal between peers. PoA algorithms rely on a set of N trusted nodes called the authorities. Each authority is identified by a unique id and a majority of them is assumed honest, namely at least N/2 + 1. The authorities run a consensus to order the transactions issued by clients. Consensus in PoA algorithms relies on a “mining rotation schema”, a widely used approach to fairly distribute the responsibility of block creation among authorities. Time is divided into steps, each of which has an authority elected as mining leader

### Storage

At least we need two different places where store our data.

We need a ledger where we store the meta-info related to a certain resource. We use PoW to save news, Ethereum proof of stake (Casper) can be tried but it is quite new and can give advantages in terms of performance but in our approach, we are interested in integrity, so security aspects. We just save the meta-info block, where due to hash we easily retrieve the whole resource. Using in the first block Ethereum as main example of architecture, we may define the ledger in the same way.

The second obvious need is a DHT (Distributed Hash Table) where save the resource. We need a different place for storing the whole resource otherwise the size of our blockchain will grow faster and faster and the validity phase will be infeasible to realize. The database has to be hash based because the hash of the resource is used as key and only GET/POST operations are performed, there is no need of range queries. A classical NoSQL DB key-value based can be Amazon Dynamo.

The need of a third database is caused by the fact that in order to enable the communication between the first and second block of the blockchain, we cannot imagine that a news definitively checked can be saved instantaneously on the first ledger. It causes that if we want to use just one ledger, or we need to add a kind of certificate attached to the meta-info to speed up its save, we cannot imagine a public permissionless blockchain as first layer otherwise, we need to make a news rerun the consensus protocol in the first layer so, we are not sure that the transaction will eventually be saved.

Due to all those considerations, we need a second ledger, it can be similar to the first, in terms of structure, but the main differences are the user rights: the first one is publicly available both in read and write mode instead, the second one is publicly available just in read mode, due to the fact that we use a private blockchain so only trusted entities can create a block.

### Hybrid

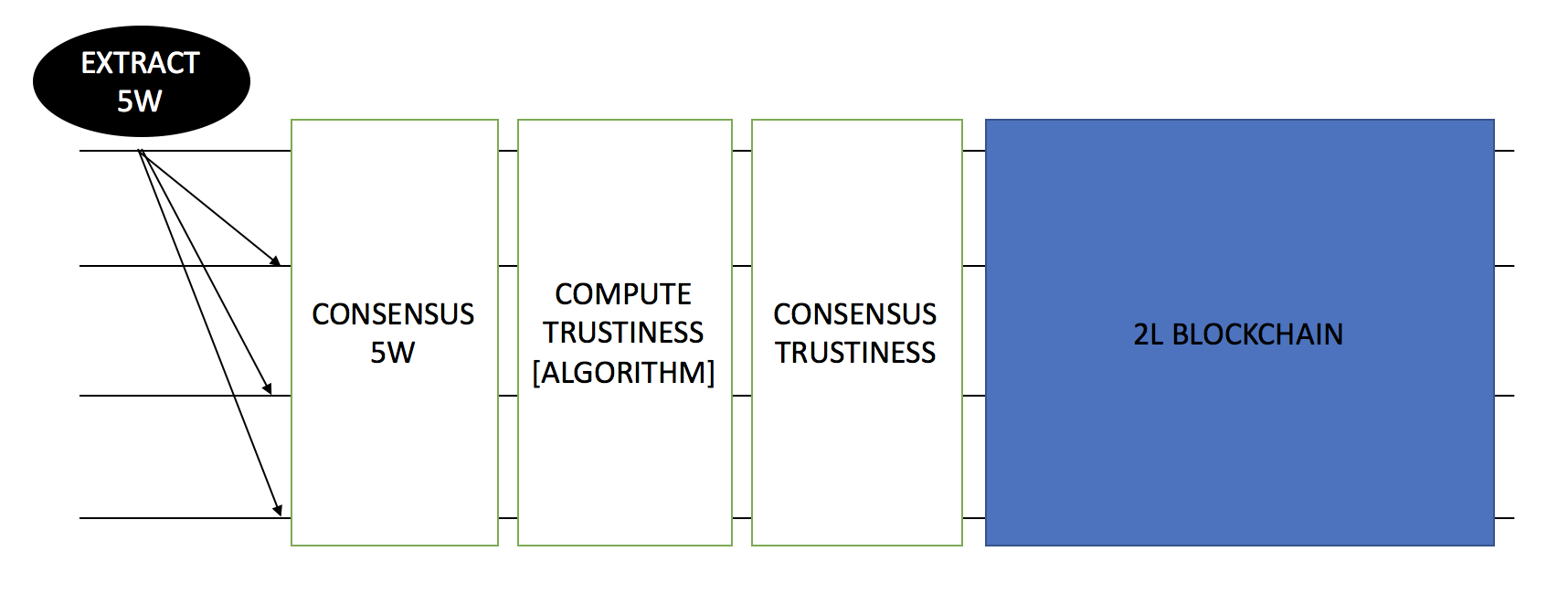
We have detailed that we use PoW in the first block and PoA in the last layer, however as we have discussed while describing storage we may think to use just one ledger. Using this approach PoS is the only method that can be used.

Switch from PoW to PoS may provide the befits like energy saving and security, in fact we have a safer network as attacks become more expensive: if a hacker would like to buy 51% of the total number of coins, the market reacts by fast price appreciation. Casper will be a security deposit protocol that relies on an economic consensus system. Nodes (or the validators) must pay a security deposit in order to be part of the consensus thanks to the new blocks creation.

If the private last layer owns more than half (regarding PBFT 2/3) of the blockchain, it can directly update a news’ score.

Starting from the idea that is behind Ethereum Casper we may think to point out the importance of checkpoint and so we have a first layer based on PoW and periodically we perform a PoS to definitively check the validity of the blocks. The only problem is that the majority of stake should be owned by the trusted entities and only them could update a news. However the computational power of first layer seems not useful for this purpose and as a matter of fact there is a clear distinction between the two blocks of our blockchain. Summing up our consideration with what discussed previously while talking about storage, the two blocks as the two ledgers should be separated and no hybrid approach is going to be considered.

## FIRST BLOCK



The step performed in this first block are:

* Post a resource
* Evaluate 5W [BLAC BOX]
* Consensus 5W
* Check validity 5W
* Consensus Trustiness

### Post a Resource

This is the step that triggers the whole execution. We have to decide if to save the resource on the database or to spread it to everyone and to store it at the end. Saving the resource before has an advantage in terms of performance because while performing consensus we send a smaller payload. However, saving it previously will cause that a fake news will be stored in the database because its correctness will be evaluated later.

In order to achieve a good tradeoff between performance and correctness we may adopt an hybrid approach. In the phase in which we have to agree on the 5w model of the resource we send the whole resource and the proposed 5w list, then after this phase every node has downloaded the resource and after evaluating the ground of trustiness, we save the original resource on DHT (Distributed Hash Table) and the meta-info on the blockchain if the news is new or pass a ground of trustiness. Doing so if we have saved on the database a fake news and an equal one (same hash or same semantic) is proposed, it can easily be disproved.

A last important aspect to point out is the management of duplicates. In fact, if we store duplicate news we just increase the size of distributed ledger, but we do not have any advantage. We store just one news by hash, if it is equal we discard the second. An attacker may publish a fake news using a DDOS attack, if the score will increase upon every ack the news will pass to the 2L and may be approved even if the second layer is too rigorous.

We save a news even if the semantic is the same of the previous because it is useful in order to establish a ground of trustiness, we link it to the previous due to the meta tag “claim”, realizing a sort of distributed tree ledger. Summing up our considerations and remembering the advantage of the IPFS idea we may describe a rule of thumb, strictly used for the first layer of the first block “an higher chain cause an higher validity of a news, if the father is trusted is easier to check the validity of TRUE children”.

### Evaluate 5W

Essentially before posting we trigger an API call to retrieve the 5w of the article (black box part), we evaluate the hash of the resource, so we spread to everyone the meta-info of the hashed resource with 5w extracted all linked with the original resource.

In particular the meta-info structure prepared in this phase has all the field filled with the only exception of the “trustiness” because this value will be evaluated only at the end of the first block.

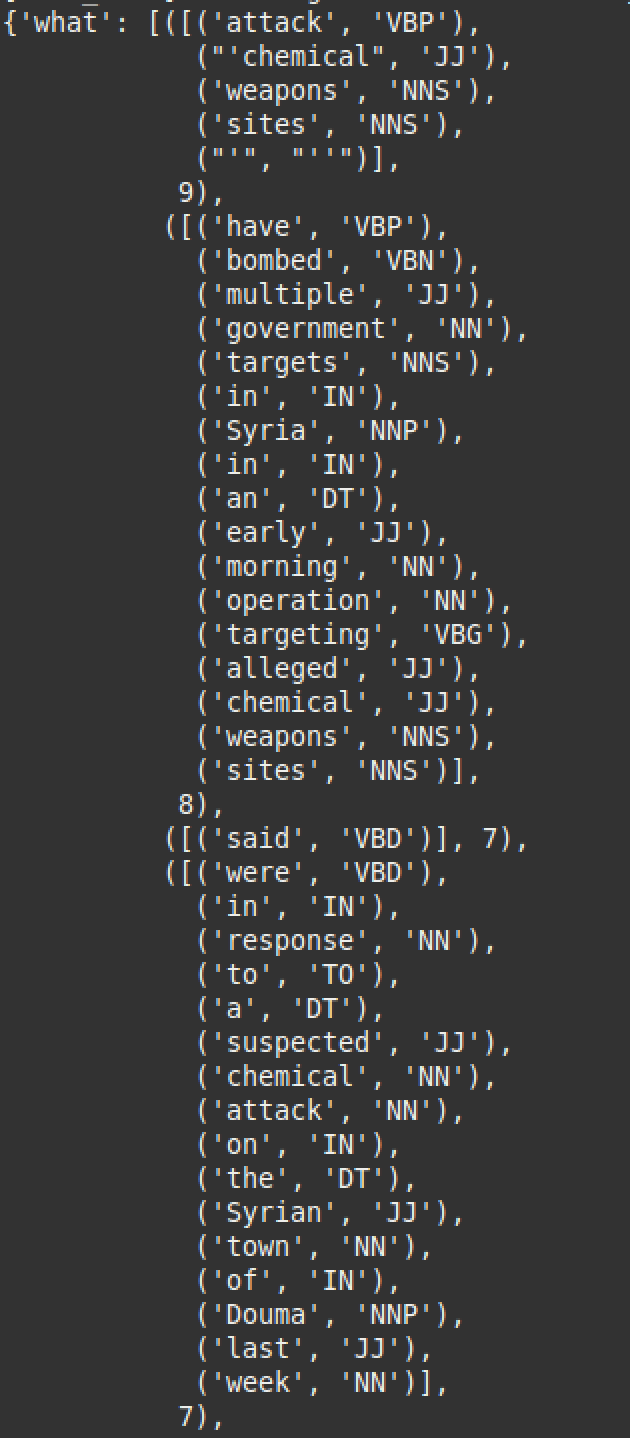
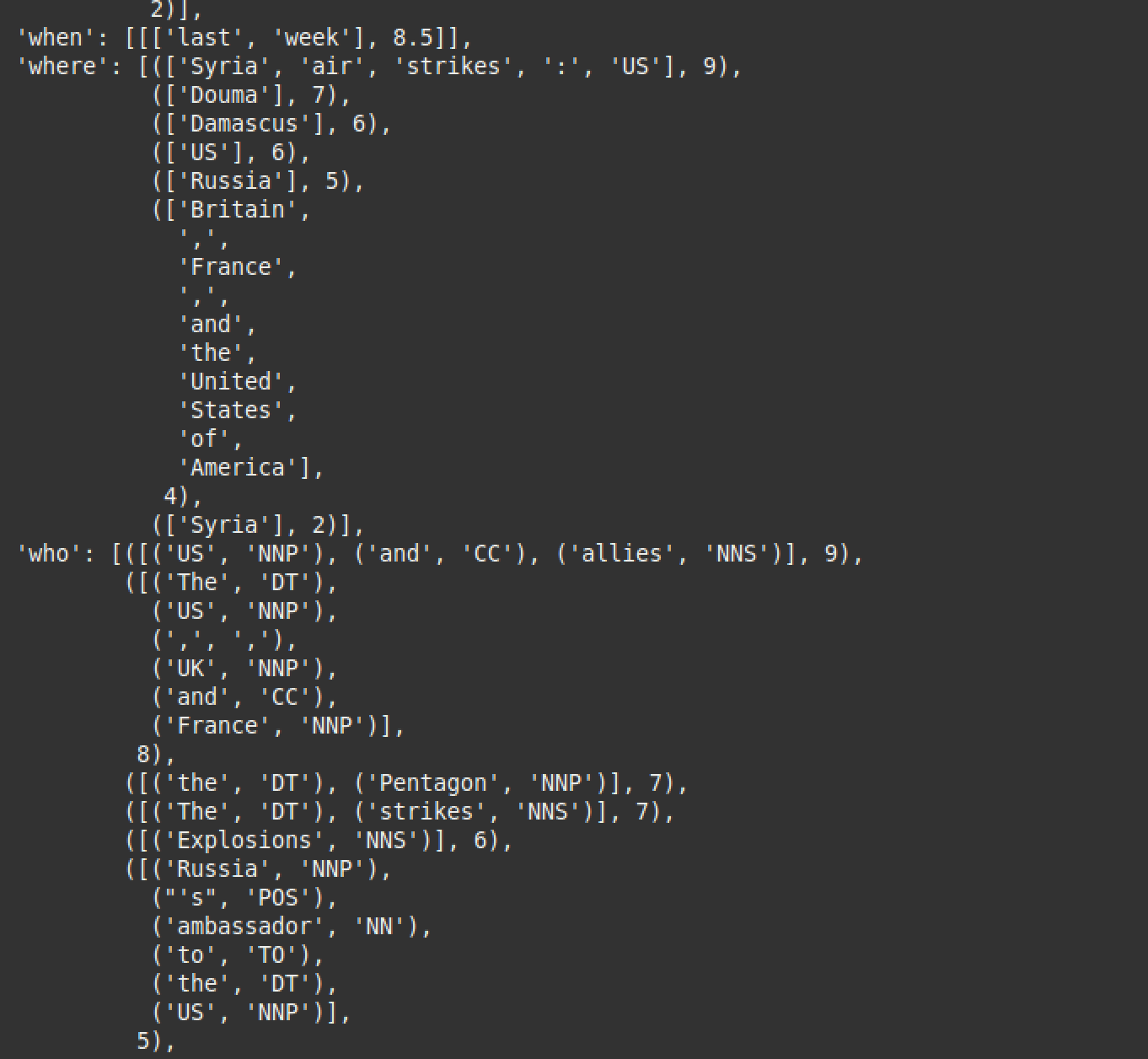
Being this phase modeled as black box, we have to discuss which API can we use, in particular we can discuss two of them:

* <https://github.com/fhamborg/Giveme5W.git>
  + This is an open source project concerning with the extraction of the 5W in a text
  + Available tutorial on main site to make it work, it works perfectly with Linux
  + Site of owner <https://www.isg.uni-konstanz.de/people/doctoral-researchers/felix-hamborg/>
* <https://dandelion.eu/semantic-text/>
  + It is an entity extractor
  + For our purpose we need to extract 5w (entity but correlated to a time and an activity)

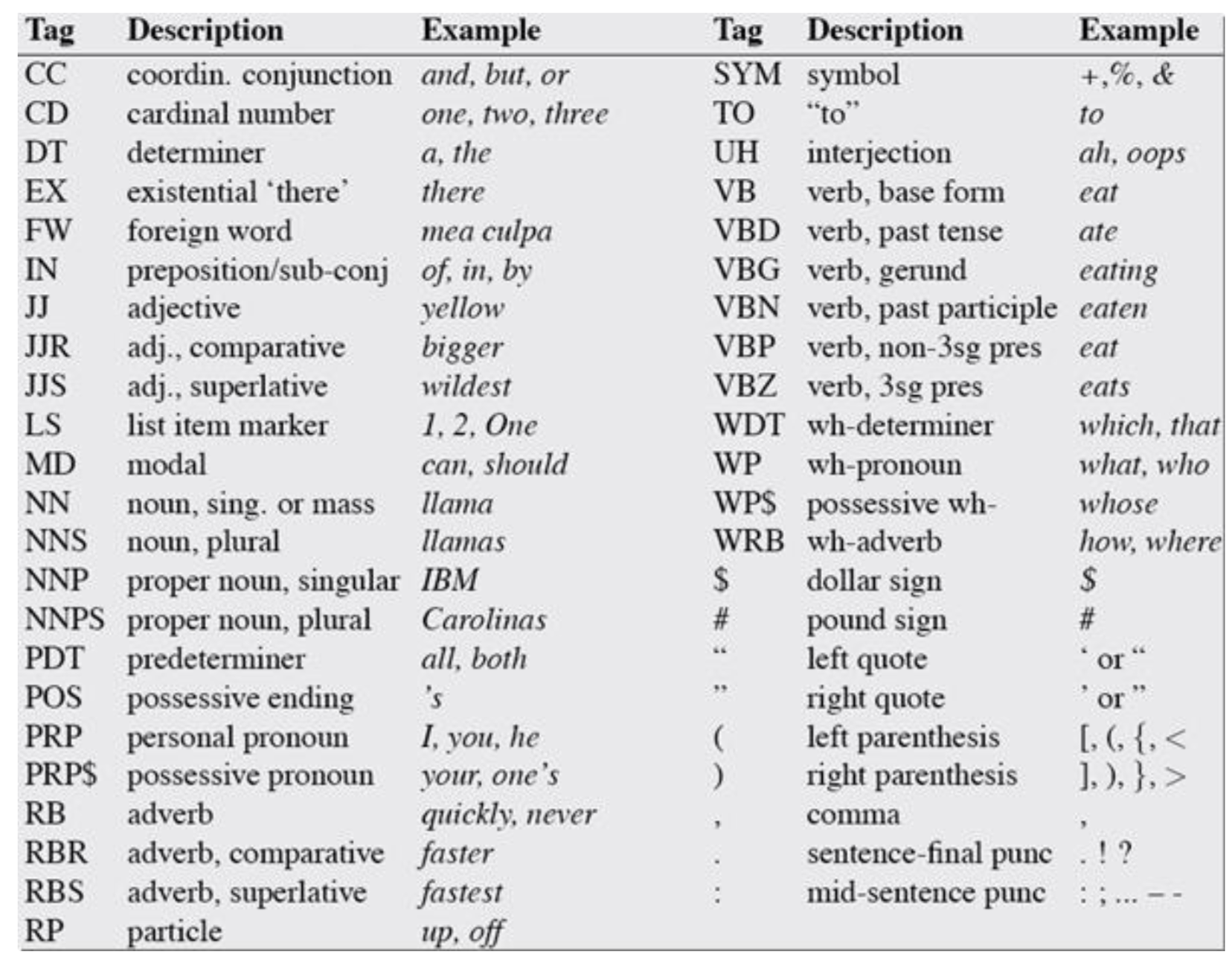
We are almost interested in the first project, due to the need of 5W extraction, the output works in the following way:

* WHAT & WHO: Sentence and score, in particular we have a list of sentences and in each one is pointed out the syntax of each word. The score of a sentence in WHO list matches the WHAT and so on.
* WHERE: It works in a similar way as previous but we do not have syntax
* WHEN: it tries to convert date if possible in datetime format due to context (understandable by machine, ex. Friday = 1220227200)
* WHY: in this example is empty, but generally can be avoided or being another sentence in can be split in remaining 4w parts and being reconsidered, however this kind of work has to be described better in a 5w implementation.

Let us consider as example an article taken from BBC: <http://www.bbc.com/news/world-middle-east-43762251> talking about chemical attack in Syria

 .

For understanding the model, an easy description of the part of speech tagging is the following:



However this model is not perfect, it has 2 main problem with our original meta-info schema:

* It works only in English, but we may consider at the moment the limitation of the language.
* It does not present dative element, it is encapsulated in WHAT tag, but may be reconstructed due to POS tagging and dandelion API

So an hybrid approach can be: we point out the sentences using the giveme5w extractor, then we extract entities with high score probability using API. Those extracted entities are just used as optional keyword for understating the main actor involved in the text: <https://dandelion.eu/semantic-text/entity-extraction-demo/?text=http%3A%2F%2Fwww.bbc.com%2Fnews%2Fworld-middle-east-43762251&lang=en&min_confidence=0.9&exec=true> (Ex. Donald Trump, …)

One option hint can be of rearranging the sentences as a list presented in our original schema, instead of use a list for each 5W tag with a matching score.

The last thing to point out is that being this project open source, written in python, it can be edited in order to fulfill our schema’s requirements. A difficult part in our schema is the accuracy below every 5W tag, however this value is difficult to calculate in this first initial phase. We use the value given as output by giveme5w (multiplied by a scale factor to be correctly tuned) combined with level of precision of entities retrieved using dandelion.

### Consensus 5w

An algorithm should be proposed, discussing pro and cons. [TODO]

### Check validity 5w

For this step is proposed an algorithm that analyze the like hood that a certain event may happen regarding the 5w list collected at the beginning.

This step is the most important and it gives as output a numeric value between 0 and 1 and according to this value we may decide if accept or reject a news. The main operation that should be performed are:

* Retrieve the meta-info of linked resources.
* Check the semantic compatibility of the 5w model of those linked resource with the one to be uploaded.

Remembering that a resource file can be described by many 5w sentences, discarding the ones whose ground of trustiness has already been checked due to the ground of trustiness of the claimed resources we have to assure the validity of the other ones, if present :

* Check the semantic compatibility between the already checked sentences and the other ones.
* Check the semantic compatibility between each other of all the sentences.
* For each sentence check the validity consulting the ledger.

In case that there are no other sentences to check, so all of them have a claimed source, we may trigger the spread of this news to the layer 3. We will focus on later.

We can imagine a validity check in the following way. Fixed the 5w we can evaluate the probability that the sentence is true or better compatible with the rest.

The why option can be discarded because is not useful in this case (WHY can be used in a more semantic detailed analysis).

Let us consider two who A, B (B is the facultative dative one), when D, where E and what C, we have to estimate the following probability:

The first two can be not considered but evaluating the third and fourth we can say that WHERE and WHEN are useful only if are considered together (they give a kind of context) and the last one (the action) is useful only regarding the probability that a person performs some action, but given an estimation of the usual context of the main actor involved and the probability that a certain action may happen given WHERE and WHEN we can redesign the probability in the following way:

Doing the same calculus starting from Where and When joined as D we have:

… we do the calculus for A but it is the same for B …

… being … so being negligible such as P(D) …

Now we can imagine a kind of algorithm (similar to **simulated annealing** but in the reverse manner) that gives as output the first estimation of trustiness.

1. Fix a value T (the minimum validity of news we may collect, so we start by the most valid so not new) and estimate value N (Number of Doc useful remembering the total number of Doc analyzed) that is correlated to T
2. Fix the actor A and find all Where possible according to accuracy
3. Fix the actor A and find When given Where according to accuracy
4. Calculate the first probability, if the number of Doc analyzed is greater than N, stop else go deeper
5. Perform step 2,3,4 for actor B (dative) if present
6. Given the context evaluate the last 3 probabilities (like previous)
7. Multiply the total result with T and compare with a threshold to decide if accept or not a news

What is important to be remembered is: If performing step 2,3 we find something not compatible we have as outcome 0 (all results are multiplied), otherwise a value to be compared with threshold, but if we find those not compatible sentences with a lower value of T, we may decide to go on and at the end if we accept the news we must be suspicious on the never acked one.

We can exploit the first steps for A:

T = 0.8;

N = 10/T;

critical = minimum ground of trustiness

while (T > critical) {

    accuracy = 0.8

    while (accuracy > 0) {

        listWhere = list all Where by trace according accuracy

        listContext = [];

        for(where in listWhere)

            listContext.add(Fix the actor A and find When given Where above accuracy)

**evaluate probability update GLOBALLY**

        if(listContext.size > N) break

        else

            accuracy -= 0.2

    }

    T -= 0.1

    N = 10/T

}

Step 6 is based upon a similar algorithm, we proceed according to T and N, so we find the occurrences (action given A, given B, given Where and When) over the total number of sentences retrieved to estimate the last 3 probabilities.

As a matter of fact at the end we simply multiply the five probabilities, but as a more practical result (event if not strictly mathematical rigorous) we can estimate the average. In fact given two values such as 0.8 and 0.9 it is better to save the average, so 0.85 instead of multiplying, because we have an high score to compare with threshold.

To conclude this part we can affirm that the algorithm as it is presented is not efficient (all the probabilities may be evaluated in a parallel way) but it just presents the main idea behind it.

### Consensus Trustiness

First of all a consensus algorithm should be used to agree upon a value proposed by peers, so we have to discuss pro and cons.

The most important thing is that in this phase after achieving consensus the value is compared with a threshold, if this value is very low, we will reject the transaction, otherwise we are going to save within the blockchain with this score.

Now the first two layers have finished their work, so we have just to decide a checkpoint value, that can be easily be the creation of each block, for which we notify the last layer that reads the ledger ad start operating on it.

## SECOND BLOCK

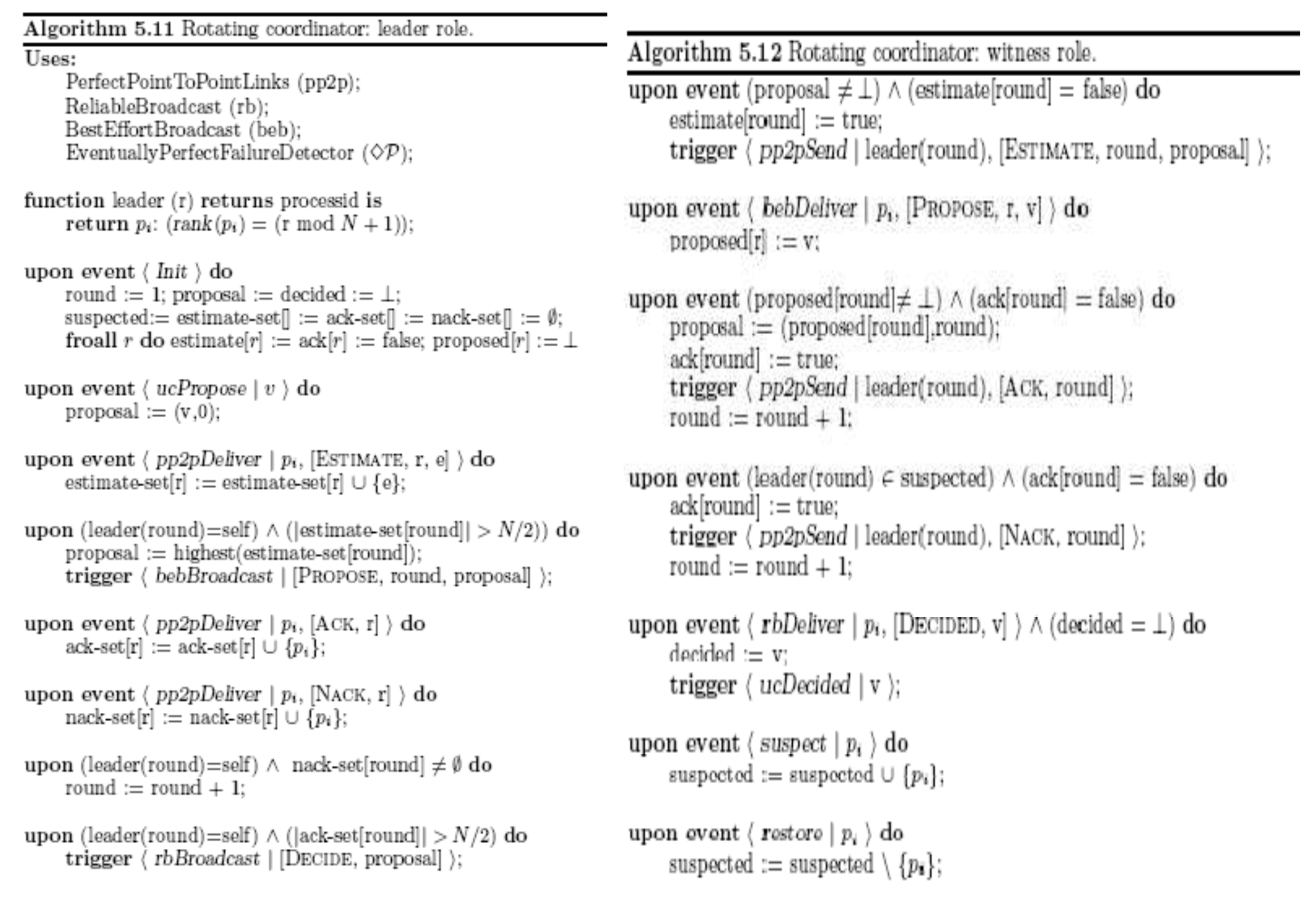
Now that a certain news has been processed in the first block and its value of trustiness is above a certain threshold, it is going to be finally checked. This block is divided in three main step:

* Evaluate dynamic witness
* Consensus
* Save of block

Given the steps, it quite intuitive that we are going to realize a “rotating coordinator” algorithm, in fact it is well suited for this layer, where we have a private permissioned blockchain that can essentially works in a partially synchronous way.

The amount of witness required for quorum is dynamic evaluated considering that it should be proportional to the value of trustiness we receive as output after first block, or better, the higher is this value, the less is the quorum needed. We can easily describe it in a mathematical approach as . K is the parameter to be tuned, it is going to be empirically evaluated and as a matter of fact it depends even by the number of peers currently available in the private blockchain layer.

Concerning with consensus we imagine a rotating coordinator algorithm:



This is a basic algorithm, it can be improved concerning with our context, in particular the function leader can be different and not only be based upon all the peers.

The final step concerns with the save of a block. In turn a leader is elected and he remains leader until the final confirmation of a news, then, when the size of the news reaches the size of a block the leader in charge is going to create the block.

### Revisited Mining

The most innovative concept behind this layer is how is considered mining. First, we should say that the whole block is considered as a black box one, we are going to consider the main ideas behind it, so we are going to define how consensus is going to be achieved and how witnesses are going to be evaluated, but we are not going to define the way in which a news is going to be checked.

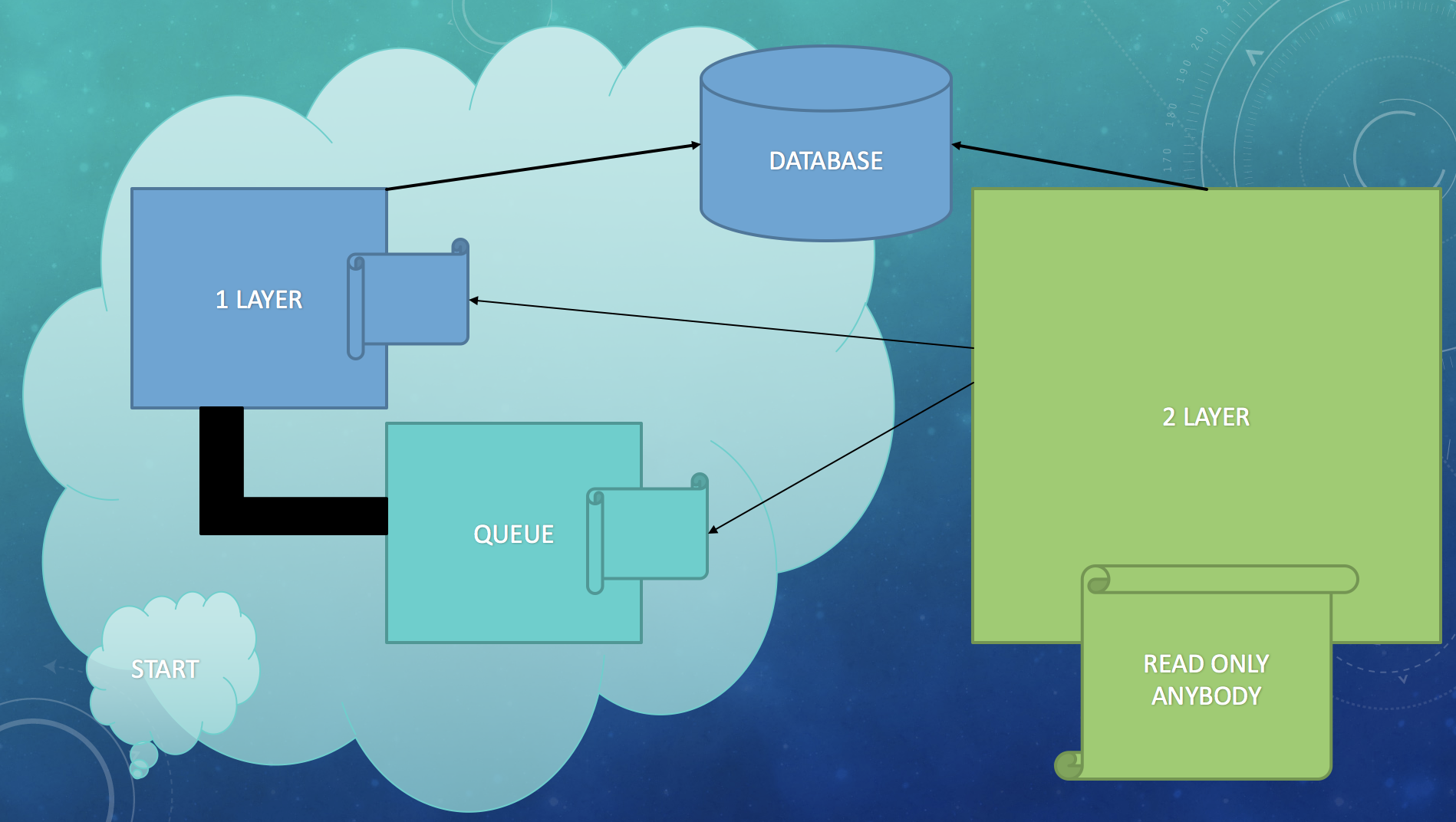
Each node can even be composed by physical people that downloads the article and manually revise is. Otherwise a certain trusted company can even propose an own algorithm and use this one to validate the news. The great idea is that we present a private blockchain with trusted entities but we do not force them to use a certain validity check method, because at the end they will promote an article and so their name will be their sort of revenue.

Miners do not earn on transaction, like cryptocurrencies, their main purpose is that their name will be an output for the news. In certain sense, they may bet on the validity of a news and use their name as guarantee. Afterwards the news with the name will be publicly available and they can monetize as they usually do.

Obviously, they can even be the promoter of an article, so we can refine this last layer not only to check the validity of anonymous news, but even for checking news published by one of the trusted entities. Considering the first layer public and permissionless it means that in this case this article can only be present at this level, as a matter of fact the amount of witnesses is static, maybe can be dynamic but in this case proportional to the source. However, this last point can be seen as a future work, for this moment it will not be deeply analyzed.

# Implementation

Concerning with the architecture we have described in theory how should work, we can rearrange the following schema :



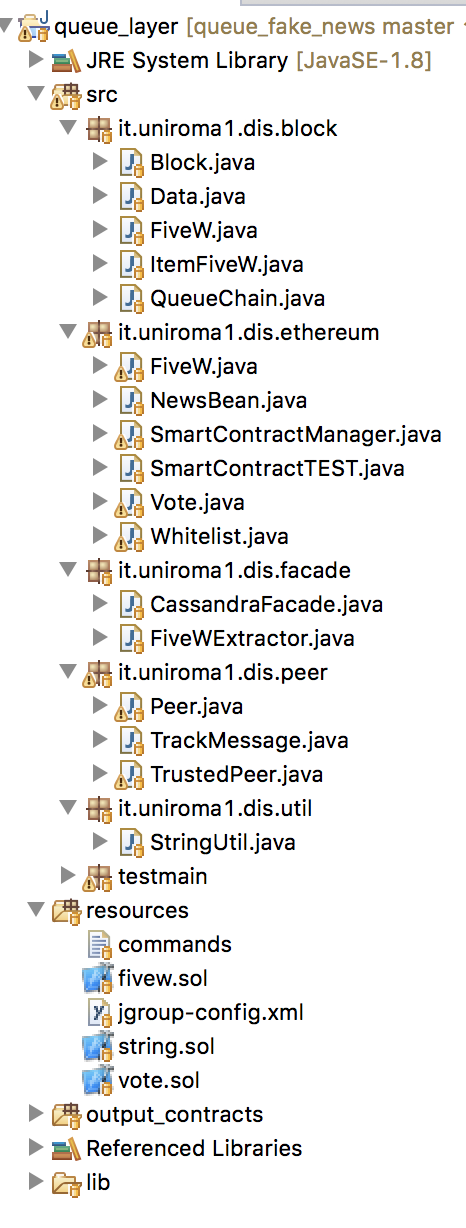
Two main actors are involved: in the first layer we have a generic “Peer”, instead for interacting with the second layer we have a “Trusted Peer”, a one that needs to be authorized.

Concerning with the main implementation choices we have:

* Start Point : An easy HTML Page or a Java Swing layer that interacts with the queue.
* Database: Cassandra, a distributed key-value based database available open source (one of the main reason for choosing it instead of Amazon Dynamo)
* Queue: It is implemented in JAVA with the use of “jgroup” for implementing a distribute messaging system between peers.
* First Layer: A smart contract written in Solidity and deployed on the Ethereum blockchain.
* Second Layer: Initially proposed Hyperledger Fabric as one of the best platforms for the creation and deployment of permissioned blockchains or Multichain but, due to some restrictions found while developing, has been presented a smart contract deployed on the Ethereum blockchain but whit a white list for the operations.

## Project Structure

The whole project is organized in the following way:



Under “src” folder we have five main packages and one just for testing the whole architecture:

* Block : Blockchain structure and data-mapping for meta-info.
* Peer : Peer structure and method to be invoked for implementing the “Zyzzyva lucky case” with the help of TrackMessage, that is just a message encapsulation. We have the TrustedPeer instead that is whitelisted and may interact with the second smart contract (the private one)
* Util : Common operations such as conversion of string to byte.
* Façade : It integrates the external modules’ interfaces, or better the python module concerning with the 5w extraction and the Cassandra driver for POST/GET operations.
* Ethereum : This package describe the java classes that represent the smart contracts deployed on Ethereum Blockchain (public and private one). We have even a test class, a manager that can be invoked by peers to call smart contracts’ interfaces and a bean representing the main fields of the news structure defined on the smart contract.

The “resources” folder contain the two smart contract (fivew public, vote private) and another one used as util dependency dealing with strings. The output of the compiled smart contracts is in “output\_contracts” folder. In resources we have also a configuration file for the jgroup library used by peers for exchanging messages and a set of commands for compiling the smart contracts and generating the java interface classes.

## Start Point

The HTML/Java Swing page has to interact with the main architecture, so the input may be directed to the queue or to the first layer. Being the queue connected to the first layer we may even think to leave the decision up to the user, but the user does not always choose the best.

Deciding to tart on first layer, after ”5w consensus” we can check the rule of thumb (due to a smart contract) and if satisfied go to queue. Essentially the queue works only as a repository, there is no logic.

Starting from the queue instead make us think about “lucky case”. The user extract 5w and send to peers without trustiness value. Then is performed consensus on this field (with reference to acked sentences and rule of thumb). Peers can ACK or send a NACK concerning to 5w list or trustiness and so, evaluating the quorum result if it is ACK, it is saved in the queue else it is sent to first layer where the protocol starts. Choosing to work in a “lucky case” scenario, the last approach is discussed and it is deeply analyzed in section 3.3 when discussing of “Zyzzyva Lucky Case”.

## Cassandra Database

Concerning with this implementation, we have realized a single table with two attributes just to avoid to save the full payload of the news in the smart contract causing an high size of the full ledger upon validation. We have defined a key space “news” and inside a table “article” :

CREATE TABLE news.article (

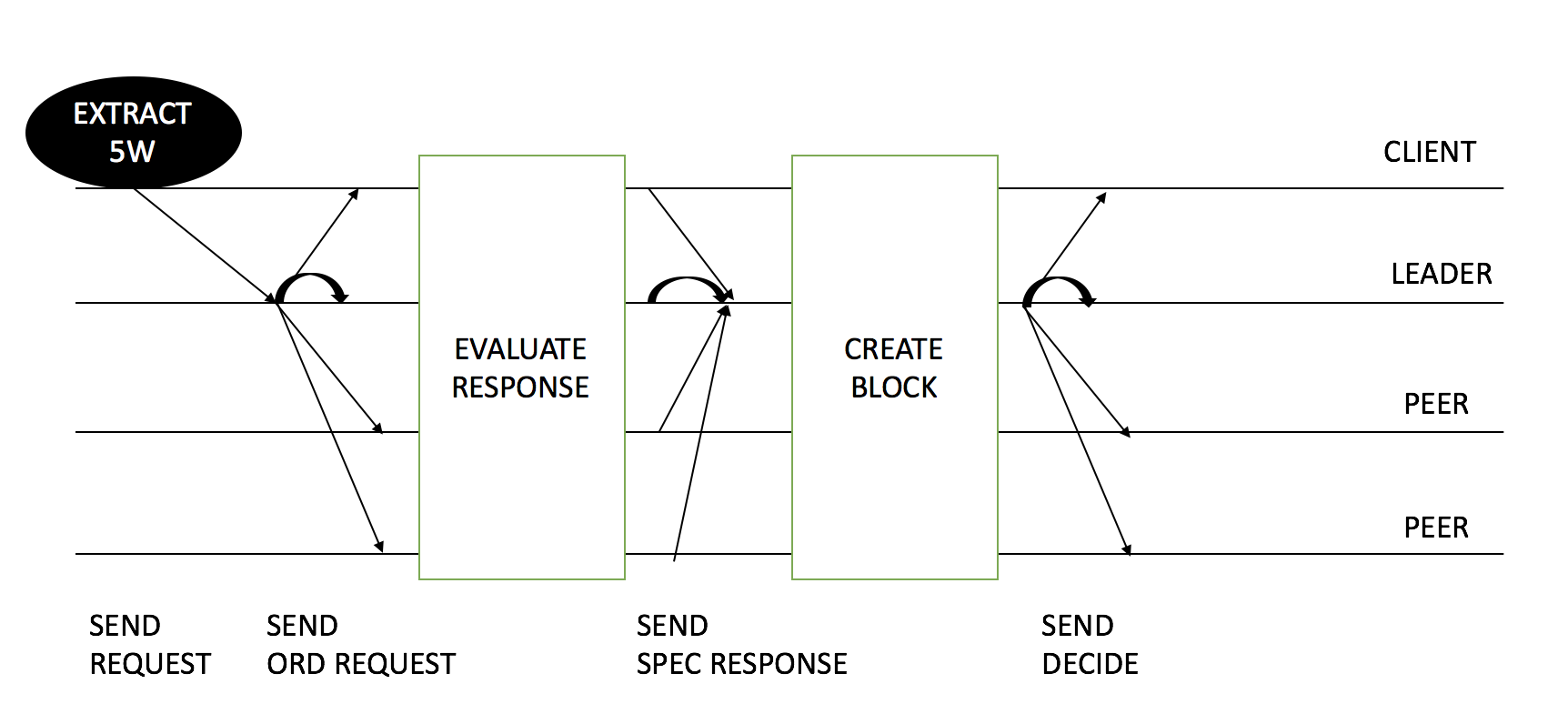
**id** text PRIMARY KEY,

**file** blob)

The integration with java code is quite easy. We have just to download the official drivers and use its interface class in order to define a “Query Statement” and then execute. In our project has been defined the CassandraFacade class that can be used to perform query on the database node. Being our main interest the GET query, a hash-based solution can be seen as the optimal one.

## Queue

What is implemented is just a set of exchanged messages dealing with the “Zyzzyva lucky case” protocol:



1. CLIENT (as peer) sends request to the LEADER

*<REQUEST, resource, name, timeStamp, client\_pub>****σc***

1. LEADER receives request, assigns sequence number, and forwards ordered request to other peers

*<<ORDER-REQ, ~~view\_id~~, seq\_num, history\_digest, message\_digest>****σp****, m>*

1. Replica *i* receives *ordered-req*, speculatively executes the request and responds to the LEADER

*<<SPEC-RESPONSE, ~~view\_id~~, seq\_num, history\_digest, message\_digest, leader\_pub, timeStamp>****σp****, rep\_pub, response>*

The view is removed being managed by jgroup library, instead in step 3 is going to be added also the digest of the outcome response.

The rest of the protocol is not implemented because what we needed was a way to speed up the first layer, in fact in case of “3f +1” consensus not achieved, the request is sent to the first layer. The final step added to the protocol is just a trigger of a DECIDED message, in order to make the peers be synchronized with the leader status of the blockchain. This message is trigger by the leader that collect the speculative responses and wait until the quorum in reached for a certain amount of time, otherwise a Byzantine Mismatch value is reached. When the news has been ACKED and is going to be saved in the queue blockchain, has been chosen a trade-off between the maximum number of data in a block and time. We try to fill the whole size of the block number but, if a timeout event is triggered, we will save just the block we few data.

The two main methods that need to be described are “checkValidity” and “minePending”.

The first describe the rule of thumb that can be used to check the lucky case, in particular ww perform three operation for deciding an ACK value:

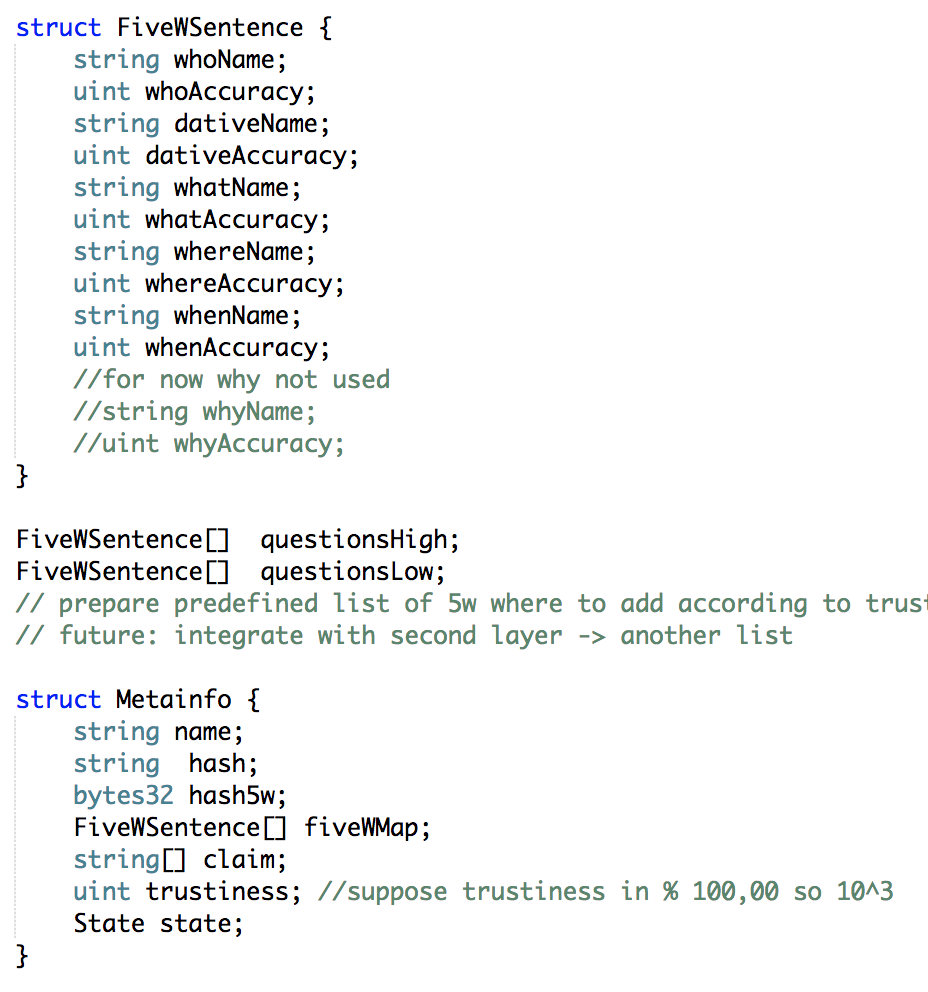
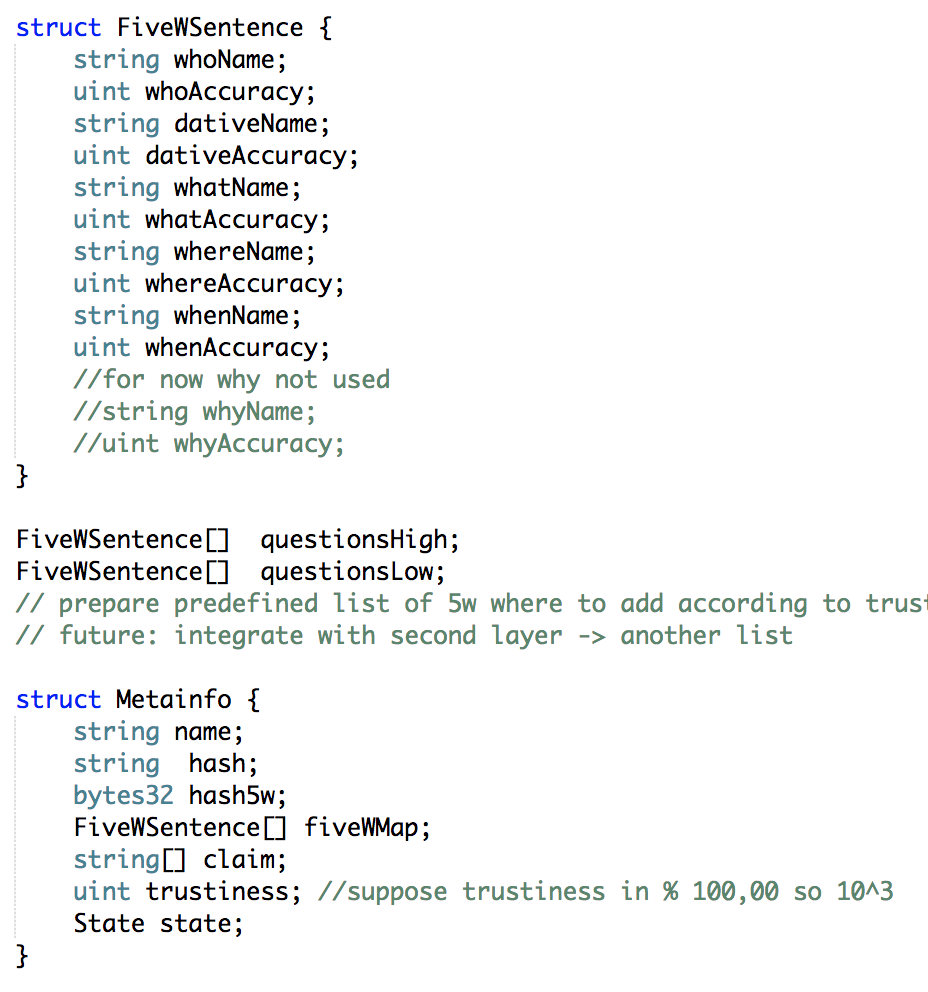
1. We check first of all if the 5w extracted match with the one we are going to extract given the payload.
2. We check if there is already a news with the same hash on Cassandra Database, in this case we return “DUPLICATED MESSAGE”
3. If all the steps are passed, we want just to check if there is a news on the blockchain that have the same 5w list

The second method describe the operations that need to be performed to achieve consensus on the 5w extracted given the payload. The peer is going to retrieve the list of payload that need to be acked, due to the first smart contract’s interface “getNewsToCheck”, given the payload extracts the 5w and then send the extracted 5w to the interface “add5w”. As a matter of fact a peer can ack the fivew just one time, so he defines internally a variable representing the number of the last acked payload.

## First Layer

The first layer is public available both in read and write mode. It is implemented as a solidity smart contract deployed on Ethereum.

Data are encapsulated concerning whit the definition of two main structures:



So the list of all the resources is public available in a mapping structure that can be read by anybody due to the interface “news”, accepting as parameter the number of the news we are interested, but we can easy access all in loop due to another public variable “newsLen”. The other important public available interfaces are: “startFiveW”, “getPayload”, “add5w”and “isHash5wPresent”.

The first is triggered upon a news is going to be inserted in the first layer. The metadata of the news are sent as input and saved in the list “news” setting as state “Newly Created”, this state will be updated later, instead the payload itself is saved in a list where all payloads whose list of 5w have to be acked, and it is deleted upon the consensus is achieved.

The second interface is triggered in order to receive the payload whose list of 5w have to be acked. It is invoked by the peer in “minePending” method (see section 3.4), and upon reading the payload the extracted 5w is sent due to the third interface.

In this third interface is first of all checked the consensus, upon achieving it the state of the news is updated and is invoked the algorithm to evaluate trustiness (see section 2.3.4). Given the trustiness score this value is compared with a threshold with 3 possible outputs: news with high score, news with a medium score, news with a low score. In the first and second case the news is added to the right list and its state is finally updated as “Consensus Trustiness”.

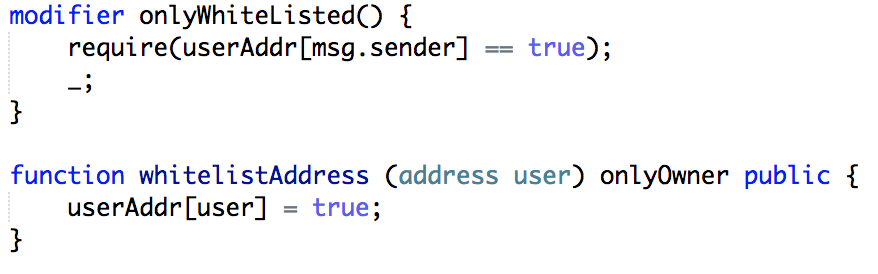
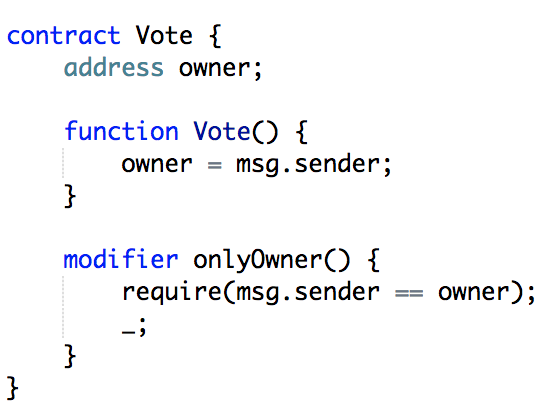
The last interface is used to validate the “Lucky case” in the queue, in order to check if a news is semantic similar in terms of 5w to another already approved one.

We have even internal functions, that essentially are util functions dealing with the import package for string management.

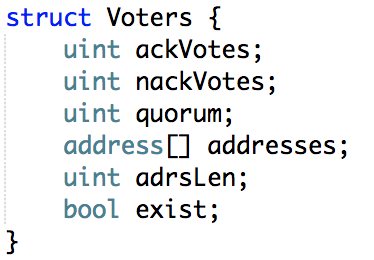
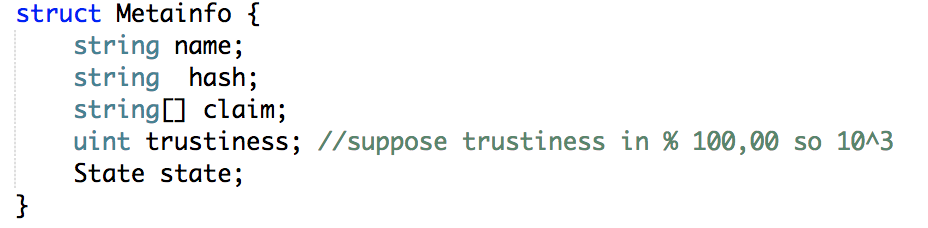
## Second Layer

This last layer is implemented as a solidity smart contract due to some assumption and, its data can be read by anybody but only whitelisted peers can access to its interfaces’ methods. The main operation are performed by the “Trusted Peer” and deals with checkpoint and final validation of a news.

News validated can be accessed in a similar way of the first smart contract by anybody, knowing the interfaces “news” and “newsLen”. The interfaces deals with the two main operations discussed previously and another one in order to add a new address in the whitelist, but this operation can be done only by the current leader or better, the owner of the contract.



Before talking about the two main interfaces, we have to discuss the main structures used:



The first one is similar to the one present in the first layer smart contract, but it does not have the list of 5w due to the fact that they are not useful in this layer, so we may save storage and even costs. The second one is used in order to perform consensus on the news by every trusted peer, is just a struct for saving the user’s address that have voted, their vote and the dynamic evaluated quorum.

Concerning with the two main interfaces we have: “addNewsToCheck” and “checkTempNews”.

The first one is invoked by the trusted peer is order to perform the checkpoint. Upon collecting the news from queue and first layer, the trusted peer triggers this method passing the parameter to fill the struct, as a matter of fact the payload is not sent because if a news passes the lucky case in the queue or the algorithm in the first layer, its payload is saved on Cassadra and can be easily retrieved knowing its hash value. The structure’s elements are saved with the state “Newly Created”

The second interface essentially fills the second structure. Every trusted peer in fact is just going to sent an ACK/NACK value for a given resource, and if the news achieves the quorum, concerning its state is updated ad checked and a generic user can find if the news is acked or not.

# Main Use Case

After analyzing the main ideas behind the architecture and the main choices done for implementation purpose, now we may describe a classic scenario involving a user that want to post a news.

There are two main starting point:

* The user acts like a peer, in this case an easy form (implemented in Java Swing) is shown.
* The user is not a peer, essentially he compiles a form on an html page and the request is randomly directed to a peer.

After preparing the form, all the rest is like a block box for the user. He may later examine the trustiness of the news published knowing the hash of the resource or easily knowing the resource.

Concerning with our architecture, the first step for the news is to pass to the queue. There we have one main actor called “Peer”. He extract the “Five W” from the resource and spread the payload to all the other peers in the queue. Every peer is going to check whether the news is a duplicated one and in case not, is going to check if other news with the same “Five W” are already present in the blockchain. This lucky case scenario, implemented as the Zyzzyva’s one, if satisfied ends with the save of the meta-info on the blockchain and the whole payload on the database implemented in Cassandra.

In case the news does not receive a majority of ACK it is sent to the first layer, realized as an Ethereum smart contract. The peer invokes the first interface of the contract “start5w”, where is prepared the structure for meta-info and the payload is inserted in a list where are present all the payloads whose “Five W” need to be acked. This operation is performed by the peer the essentially retrieve the payload from this list (“getPayload” interface) and compute the list of “Five W” sentences sending due to the interface “add5w”. On the smart contract’s side upon receiving a total number of responses for a certain payload if the majority of them are positive, is computed the trustiness of the meta-info and the is updated the status. As callback the payload of the acked news is now saved on the database.

The last part involves the second and last layer, that due to simplifying assumption is implemented as a smart contract in solidity but with a whitelist of addresses that can perform judge operation.

The main actor in this step is a “Trusted Peer”, a peer that is whitelisted. It performs the two main operation: "Checkpoint” and “Vote”. The checkpoint consist in retrieving the news from queue and first layer then adding them to the smart contract of the second layer due to the interface “addNewsToCheck”, the first operation in this interface involves the multiplication of the news’ score with a scale factor in order to evaluate the quorum. The last step instead consist just as retrieving the news from second layer, then retrieving the payload knowing the retrieved hash on Cassandra, and then vote the news due to the interface “vote”.

[ADD DIAGRAMS]

# REFERENCES

* Huckle S, White M (2017) Fake news: a technological approach to proving the origins of content, using blockchains. Big Data 5:4, 356–371, DOI: 10.1089/big.2017.0071.
* Barriocanal E, Alonso S, Sicilia M, Deploying Metadata on Blockchain Technologies. V.755, 2017, Pages 38-49 11th International Conference on Metadata and Semantic Research, MTSR 2017
* Juan Benet. IPFS—Content Addressed, Versioned, P2P File System (DRAFT 3). 2017.
* Kim, Henry & Laskowski, Marek. (2016). Towards an Ontology-Driven Blockchain Design for Supply Chain Provenance.
* Das, Amitava & Ghosh, Aniruddha & Bandyopadhyay, Sivaji. (2010). Semantic role labeling for Bengali using 5Ws. 1 - 8. 10.1109/NLPKE.2010.5587772.
* Aniello, Leonardo & Baldoni, Roberto & Gaetani, Edoardo & Lombardi, Federico & Margheri, Andrea & Sassone, V. (2017). A Prototype Evaluation of a Tamper-Resistant High Performance Blockchain-Based Transaction Log for a Distributed Database. 10.1109/EDCC.2017.31.
* Buterin, Vitalik & Griffith, Virgil. (2017). Casper the Friendly Finality Gadget.

Interesting web sites:

* <https://dzone.com/articles/blockchain-and-ai-can-defeat-fake-news>
* <https://github.com/glowkeeper/Provenator>
* <https://github.com/professormarek/traceability>
* <https://dandelion.eu/semantic-text/>
* https://github.com/fhamborg/Giveme5W.git