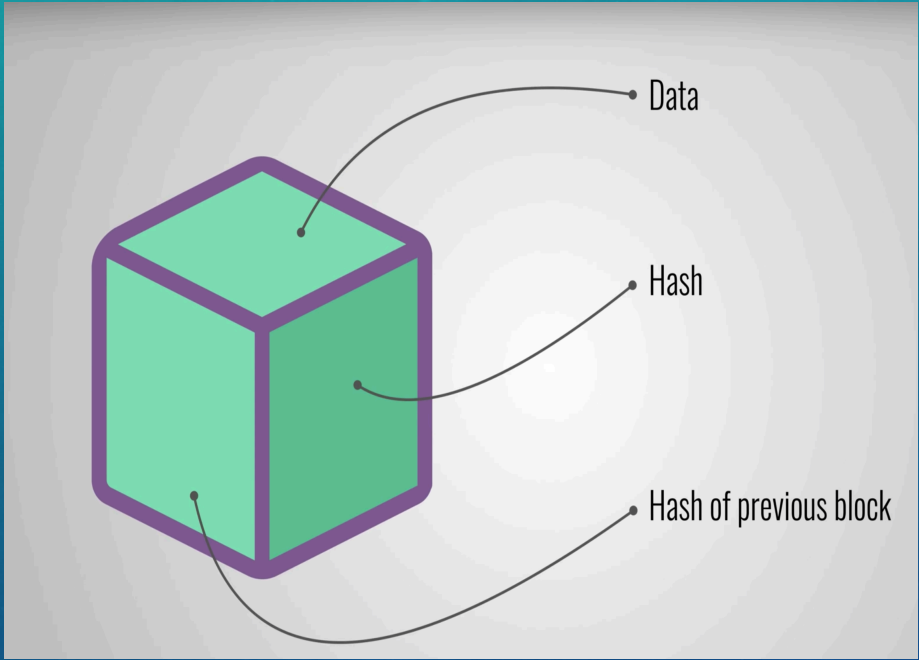
FAKE NEWS: A TECHNOLOGICAL APPROACH USING BLOCKCHAINS

ANTONIO MAURO - 1766342

BLOCK



It is possible to save data within a block, what has to be defined is:

- the way data should be stored
- the way data can be inserted or retrieved

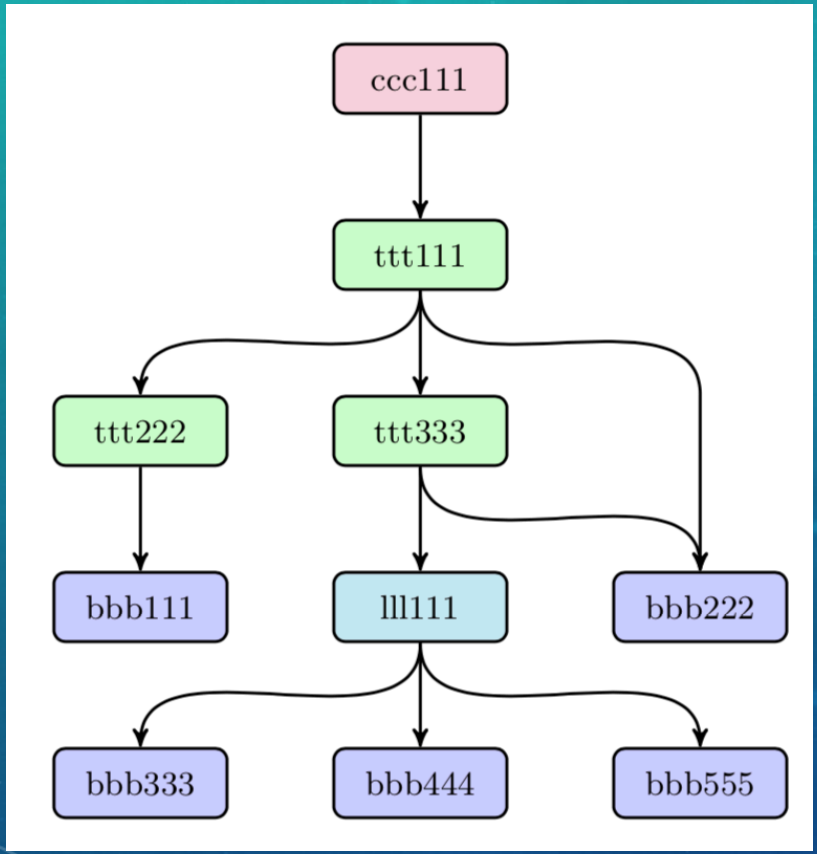
CURRENT SITUATION

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.

IPFS IDEA



- 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

Example:

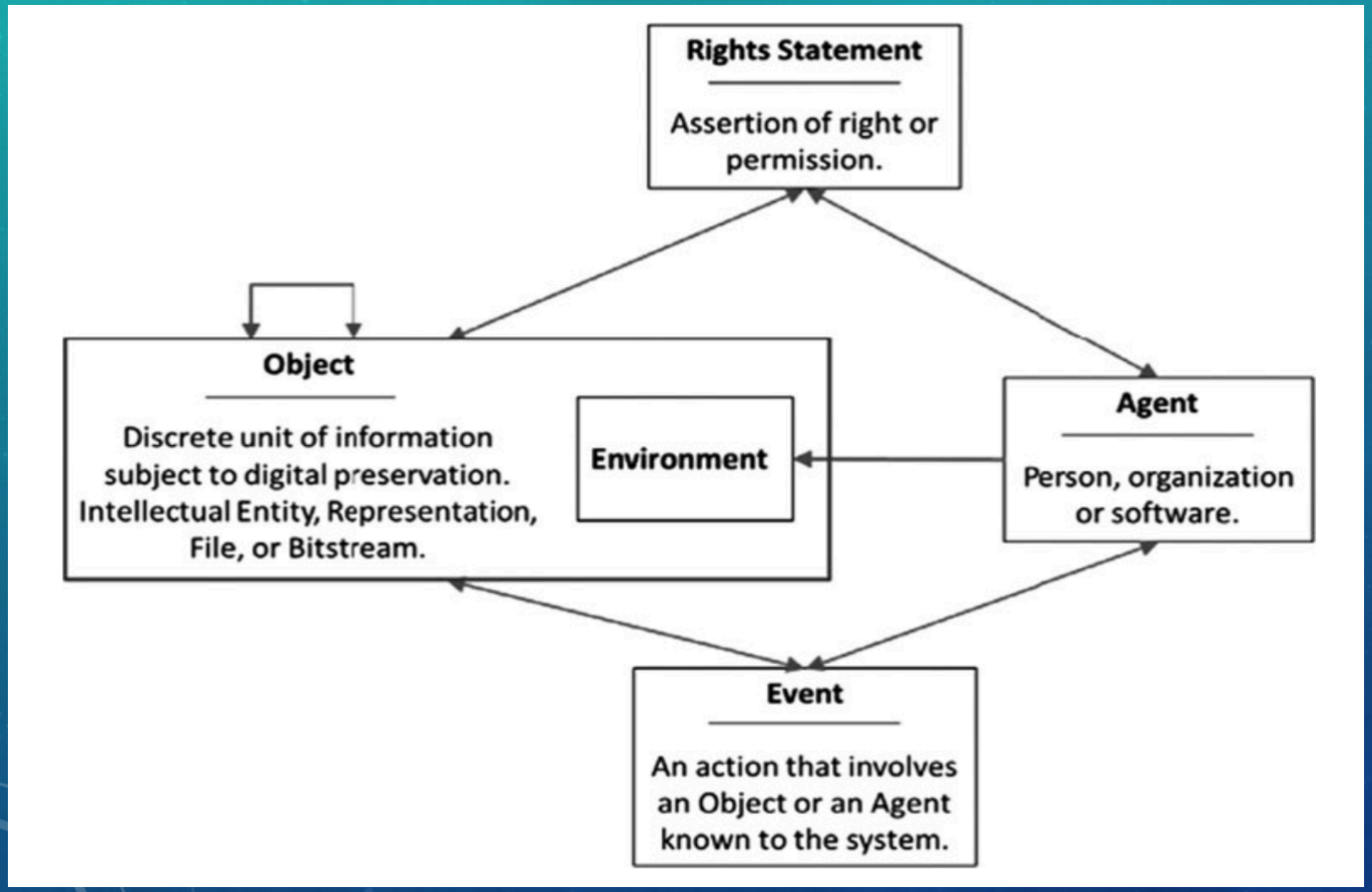
{"hash": "QmYwAPJzv5CZsnA625s...",

"service": "ipfs",

"meta":"",

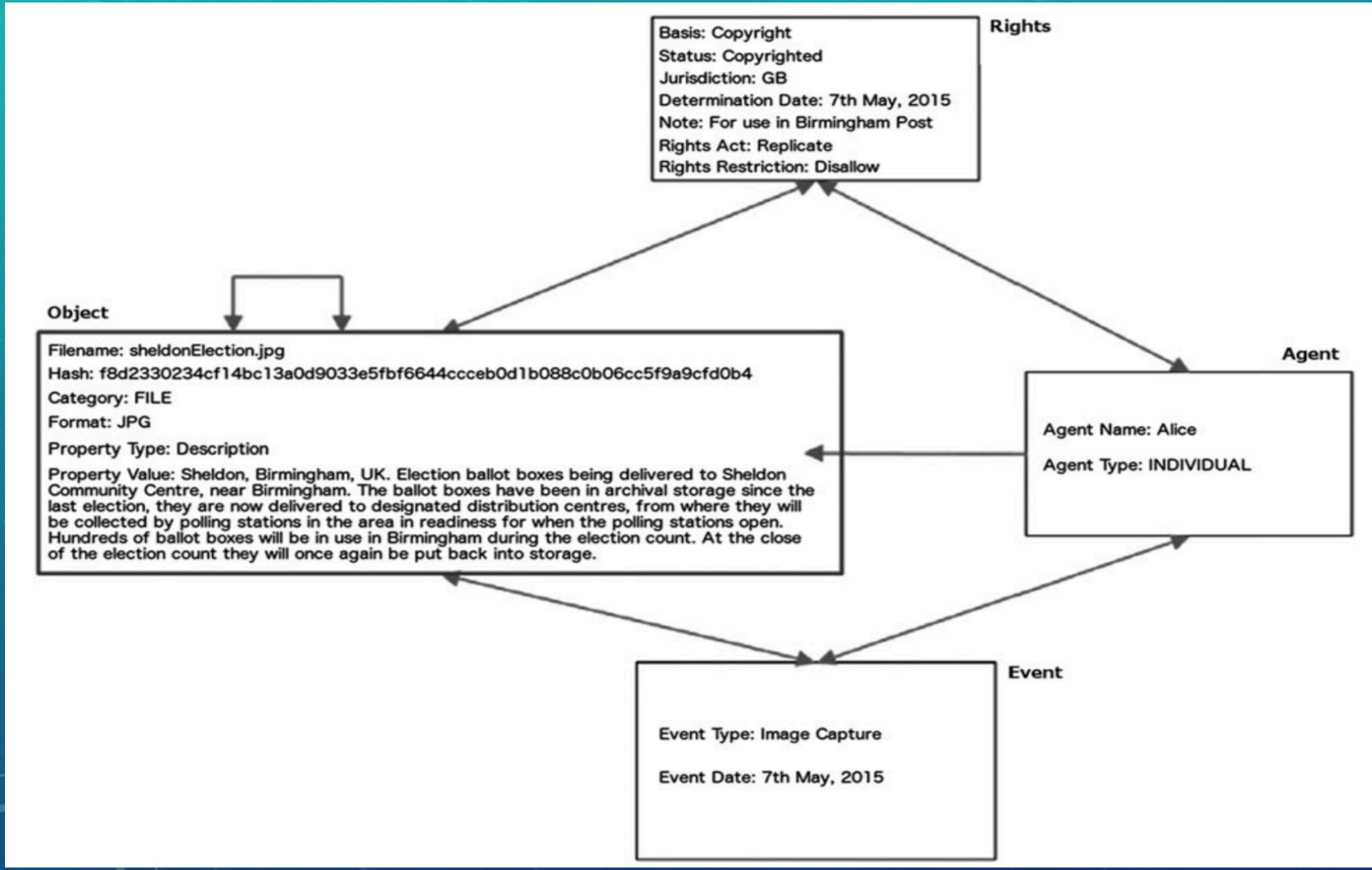
"claim":"0xbd53b39f64ce9a96d..."}

META-INFO (PREMIS)



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

PREMIS - EXAMPLE



… by retrieving the data above, Bob is confident as to the authenticity of the image Alice sent.

He can also apply edits and record information about those changes, thus creating a provenance chain for the picture.

OPERATION FOR PROVENATOR APPLICATION

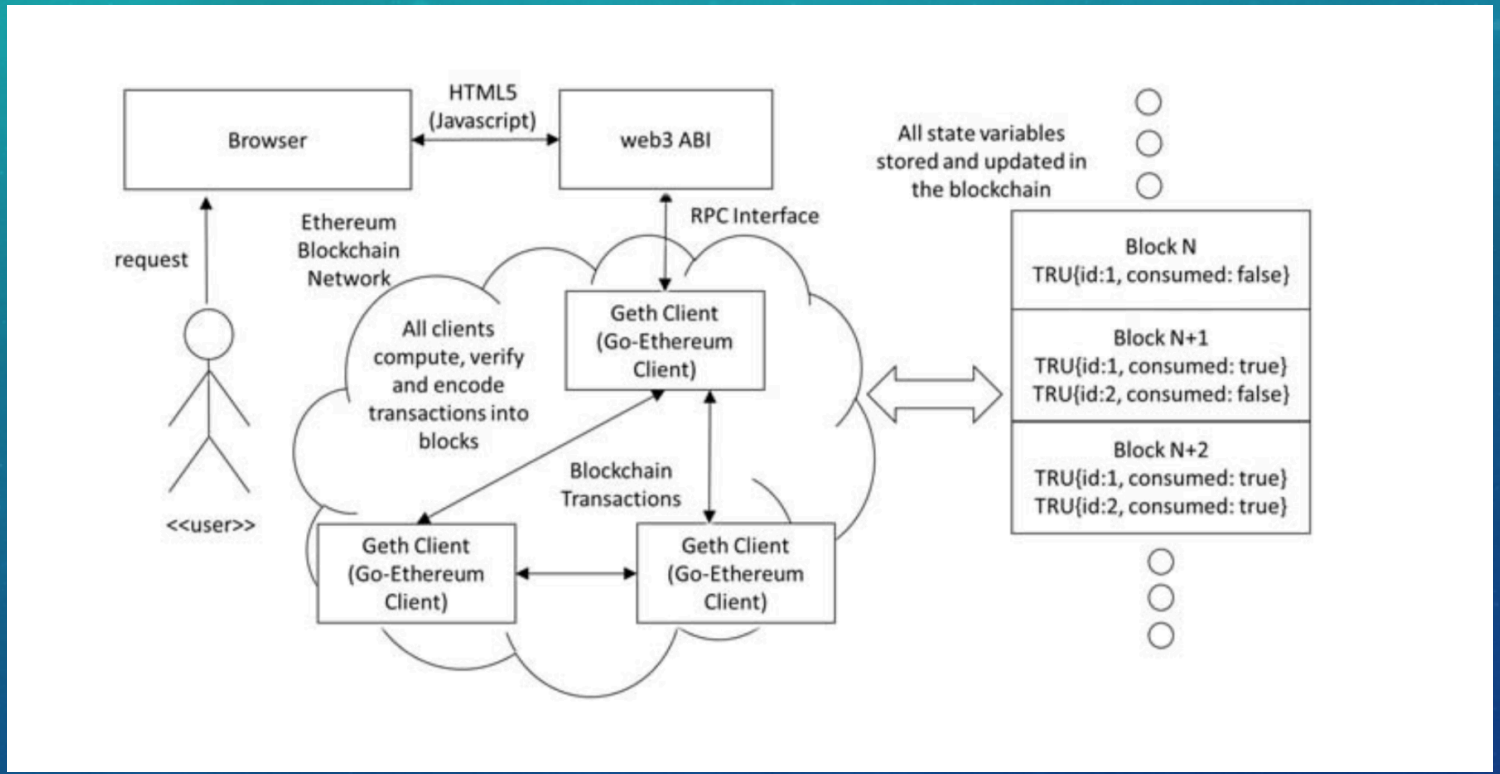
TO ADD:

1. Get a cryptographic hash of the digital media resource.
2. Create the PREMIS of the digital resource.
3. Sign the transaction that stores the cryptographic hash of the digital resource, and its associated metadata, on the blockchain.

TO CHECK:

1. Get a cryptographic hash of the digital resource.
2. Check whether that hash exists on the blockchain.
3. If the hash exists, retrieve the associated metadata.

ARCHITECTURE (1)



ARCHITECTURE (2)

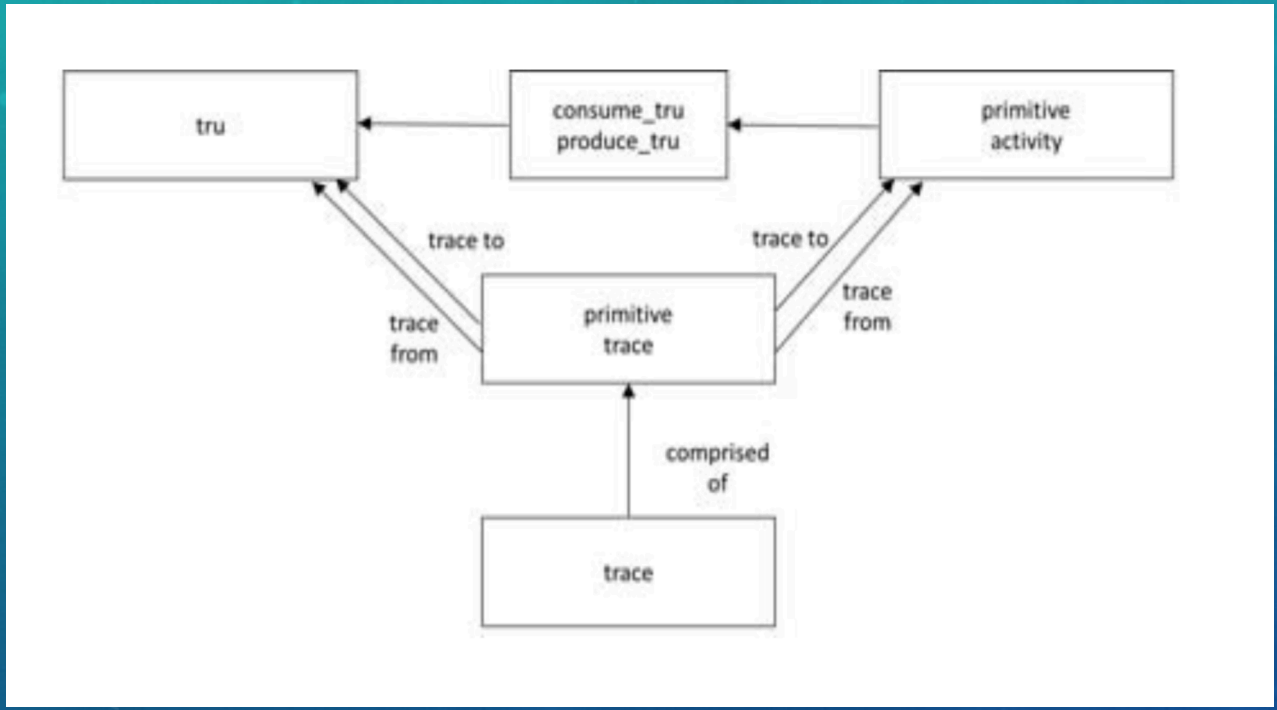
- We need a Blockchain network client (i.e. geth) is used directly or through an intermediate JavaScript/Web layer to deliver data onto the Blockchain.
- In the context of this proof-of-concept, anonymous data is aggregated and stored using a relatively simple mechanism within a **Smart Contract** on the Blockchain network, implemented using the **Solidity** programming language.
- The data is combined with expert knowledge encoded within the Smart Contract to implement the Agent Based Model used to model the spread of infectious disease.

PROVENATOR - KEY POINTS

With reference to the example at https://github.com/glowkeeper/Provenator, we have to manage:

- Ethereum blockchain (or an ad hoc designed blockchain)
- Ethereum smart contracts (save and retrieve data)
- A client interface for adding news (or check)

TOVE - SCHEMA

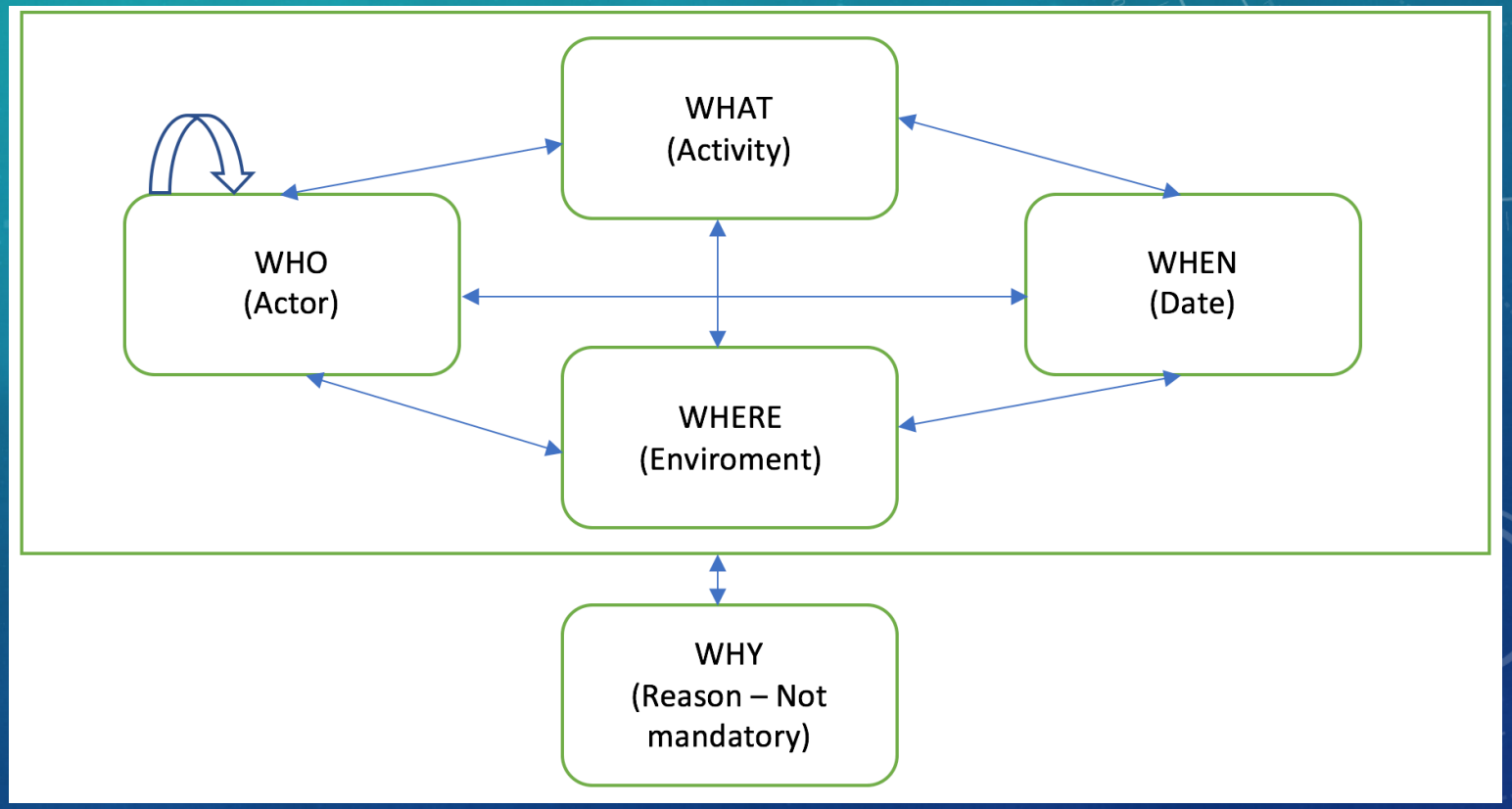


TOVE Traceability Ontology Excerpt :

- 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.
- Primitive activity is the activity representation that must be traceable, since a primitive activity is neither an abstracted nor aggregated entity.

NLP - CHALLENGES

- STORE TEXT INSTEAD OF JUST IMAGE, SAVE SEMANTIC:
- 5W MODEL (WHO, WHAT, WHERE, WHEN, WHY)
- KEY- WORDS (combined with 5w), NOT USED

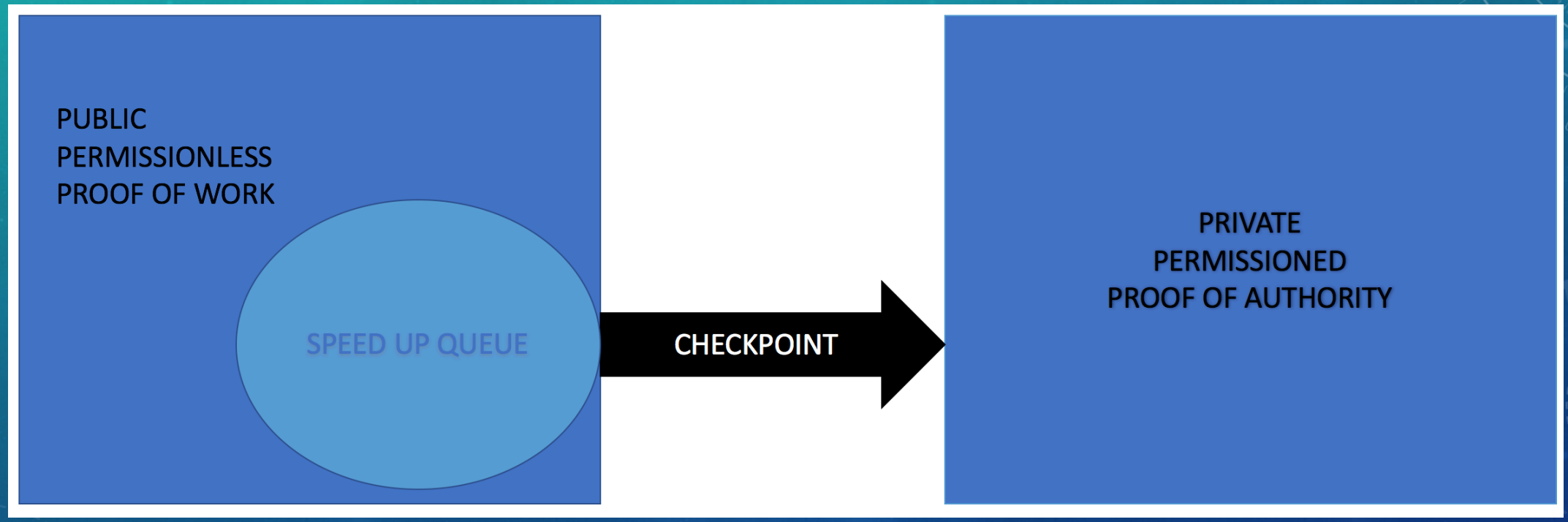


META-INFO STRUCTURE

```
{
    name: "example.pdf",
    hash: "0x0123abc",
    claim: ["hashOf1", "hashOf2"],
    trustiness: 0.8,
    rights: {
        edit: false
        share: false
    },
    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], ...
    }
}
```

Rights useful only with a private context or with signature (NOT ANONYMOUS)

2 BLOCKS


PUBLIC
PERMISSIONLESS
PROOF OF WORK
SPEED UP QUEUE
CHECKPOINT
PRIVATE
PERMISSIONED
PROOF OF AUTHORITY

FAQ - ARCHITECTURE

PROOF APPROACH

- First layer is based upon PROOF OF WORK, the architecture is similar to the provenator one, the only difference is that upon a save of a block the second layer is triggered
  - In a better analysis the second layer is triggered as CHECKPOINT
- The second is based upon PROOF OF AUTHORITY, it is a private permissioned one (the first layer is permissionless and public)
  - The concept of miner is revisited, we have private participants, well known companies involved in journalism, that has to manually judge an article (or prepare an own code).
  - Upon receiving consensus the news is guaranteed to be true and his score UPDATES THE OLD ONES (or better in saved in 2L)

- The final output of a news consist of adding a line in metainfo when the news is uploaded that is:
- “powered by Wired,….”

1 LAYER PUBLIC AND META-INFO WITH SIGNATURE ?

- Public: Everyone can post a news, otherwise we realize a kind of monopoly of information. Otherwise the news has to be approved.
- Meta-info: No need of signature of resource for many reason:
  - Security: Every news has the same initial score of another, we cannot hack an account a pubblish whatever we want.
  - The validity of the news is evaluated better in layer 2, where we have a permissioned blockchain, we just need a ground of trustiness according to the content

DUPLICATE

- 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.
  - 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.
  - RULE OF THUMB: An higher chain cause an higher validity of a news, if the father is trusted is easier to check the validity of TRUE children.

VALIDATION - WHEN

- 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 db.
- HYBRID: 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, than after this phase every node has downloaded the resource and after evaluating the ground of trustiness, we save the original resource on DHT and on the blockchain the meta-info if the news is new or pass a ground of trustiness. Doing so if we have saved on the db a fake news and an equal one (same hash or same semantic) is proposed, it can easily be disproved

WHY PROOF OF AUTHORITY

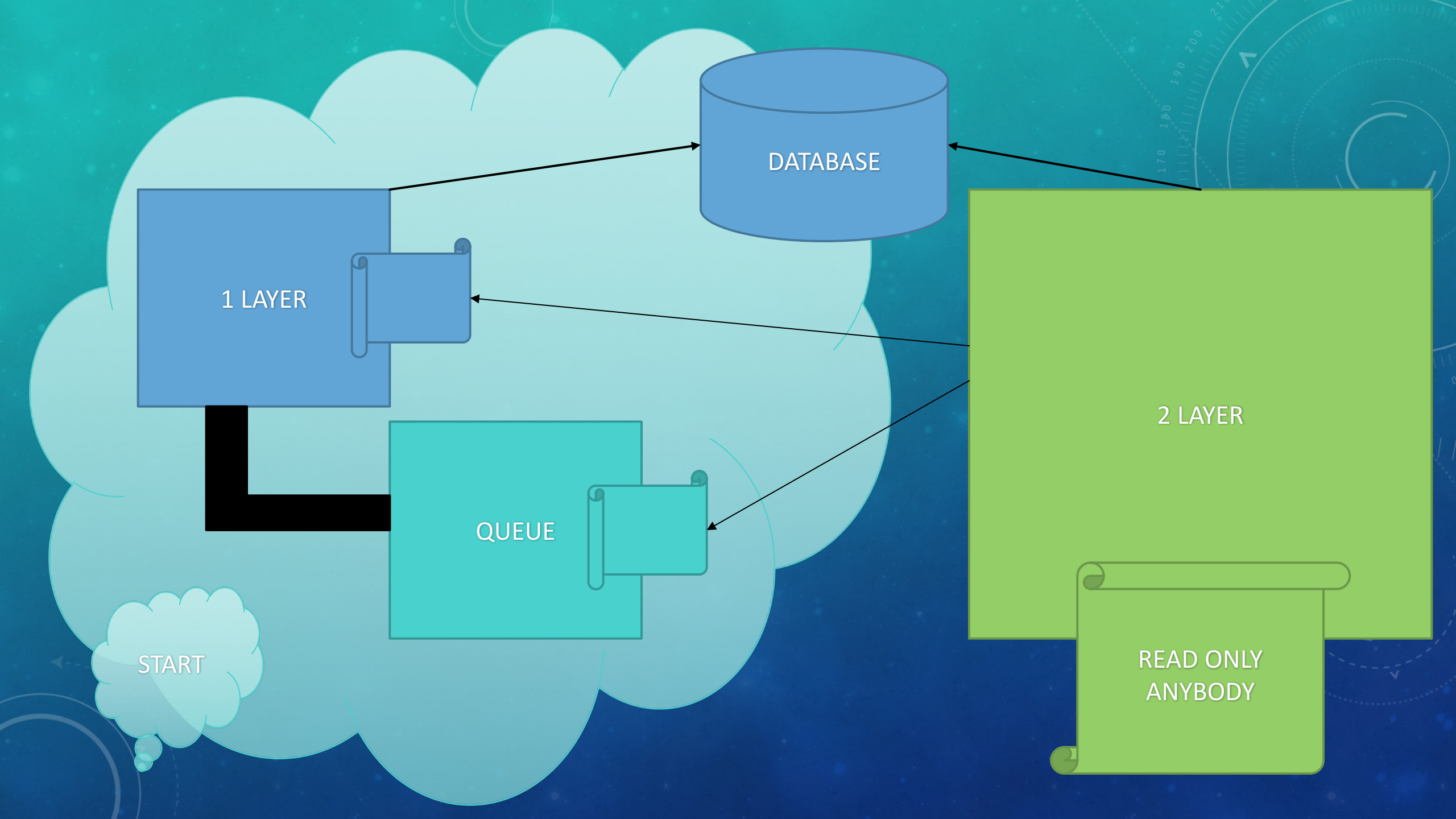
- Proof of work is already used in first layer, even if it is quite quickly depending on the size of the ledger. However it is just a filter.
- We want to be sure that a news is correct:
  - Being the second blockchain private only trusted parties can partecipate
  - PoS does not work due to the algorithm we want to use, a kind of rotating coordinator.
  - PoA is suitable for private network

REVISITED MINING

- The main concept involve the second layer where we use PoA.
- 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.
- 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 tha 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.

MORE ON STORAGE

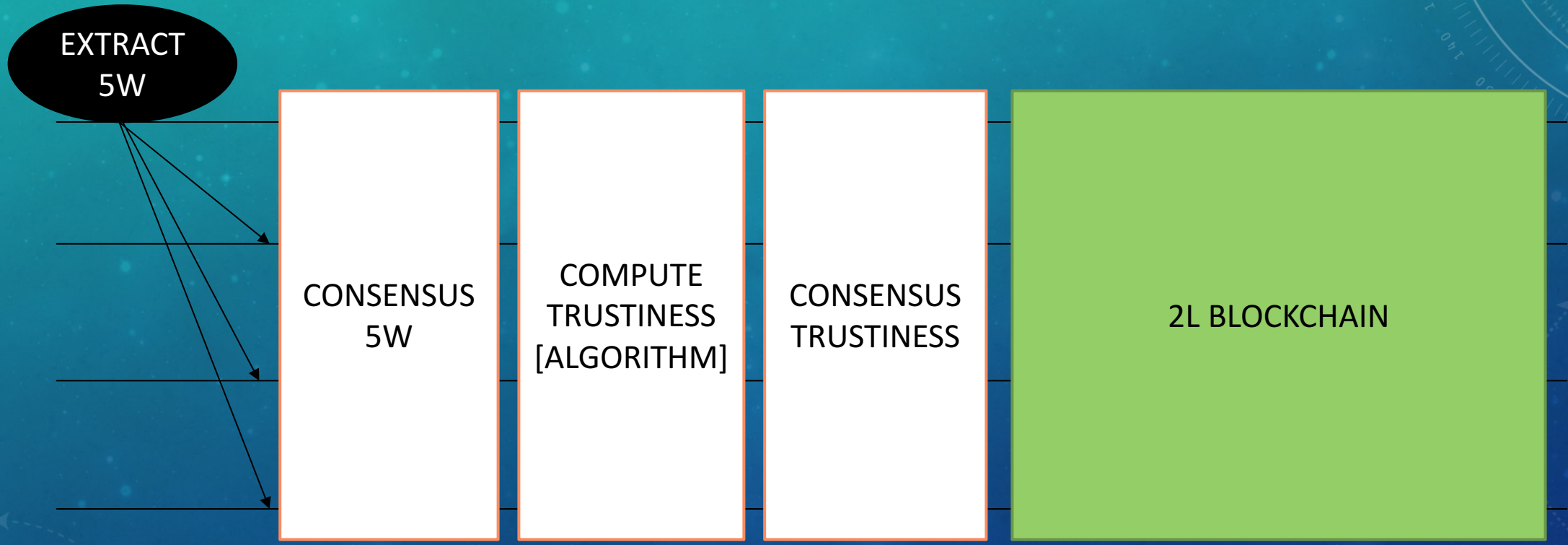
- 2 PLACES WHERE STORE:
  - LEDGER: We use PoW to save news, (Ethereum proof of stake can be tried but it is quite new). We just save the meta-info block (hash can be used to retrieve the whole resource). Can be used Ethereum as architecture so we define the ledger in the same way.
  - DHT: We need a distributed database where save the resources. It has to be hash based (the hash of the resource is the key) and only GET/POST operation are performed (NO RANGE QUERIES). A classical NoSQL DB key-value based can be Amazon Dynamo.
  - QUEUE: It is an in memory storage for each peer in order to speed up communication between the two main blocks.


DATABASE
1 LAYER
2 LAYER
QUEUE
START
READ ONLY
ANYBODY

IMPLEMENTATION – TO FIX

- DATABASE : Cassandra, key value based, distributed, opensource (instead of Amazon Dynamo)
- BLOCKCHAIN TECHNOLOGY:
- 1L -> ETHEREUM SMART CONTRACT (we want to run a smart contract in order to decide if save)
- 2L -> Hyperledger Fabric is one of the best platforms for the creation and deployment of permissioned blockchains. Other ???

SCHEMA


EXTRACT
5W
CONSENSUS
5W
COMPUTE
TRUSTINESS
[ALGORITHM]
CONSENSUS
TRUSTINESS
2L BLOCKCHAIN

API 5W EXTRACTION

- https://github.com/fhamborg/Giveme5W.git PROJECT 5W EXTRACTION
  - Available tutorial on main site to make it work. [works perfectly with Linux]
  - Site of owner https://www.isg.uni-konstanz.de/people/doctoral-researchers/felix-hamborg/
- https://dandelion.eu/semantic-text/ ENTITY EXTRACTOR
  - Works only with entity, for our purpose we need to extract 5w (entity but correlated to a time and an activity)

GIVE ME 5W – OUTPUT (1)

- The example article is taken by the BBC: http://www.bbc.com/news/world-middle-east-43762251
- The output is characterized in such a way:
  - Each sentence is labeled with an incremental counter.
  - WHAT & WHO: SENTENCE AND SCORE, in particular we have a list of sentences and in each one is pointed out the ”part of speech” of each word.
  - WHERE: works in a similar way as previous but we do not have POS
  - WHEN: it tries to convert date if possible in datetime format due to context (understandable by machine, ex. Friday = 082345..)
  - WHY: in this example is empty, but generally can be avoided. OR WE SHOULD SPLIT IN ANOTHER SENTENCE

GIVE ME 5W – OUTPUT (2)

```
{'what': [([('attack', 'VBP'),
            ("'chemical", 'JJ'),
            ('weapons', 'NNS'),
            ('sites', 'NNS'),
            ("'", "'")],
           9),
          ([('have', 'VBP'),
            ('bombed', 'VBN'),
            ('multiple', 'JJ'),
            ('government', 'NN'),
            ('targets', 'NNS'),
            ('in', 'IN'),
            ('Syria', 'NNP'),
            ('in', 'IN'),
            ('an', 'DT'),
            ('early', 'JJ'),
            ('morning', 'NN'),
            ('operation', 'NN'),
            ('targeting', 'VBG'),
            ('alleged', 'JJ'),
            ('chemical', 'JJ'),
            ('weapons', 'NNS'),
            ('sites', 'NNS')],
           8),
          ([('said', 'VBD')], 7),
          ([('were', 'VBD'),
            ('in', 'IN'),
            ('response', 'NN'),
            ('to', 'TO'),
            ('a', 'DT'),
            ('suspected', 'JJ'),
            ('chemical', 'NN'),
            ('attack', 'NN'),
            ('on', 'IN'),
            ('the', 'DT'),
            ('Syrian', 'JJ'),
            ('town', 'NN'),
            ('of', 'IN'),
            ('Douma', 'NNP'),
            ('last', 'JJ'),
            ('week', 'NN')],
           7),
```

```
           2)],
'when': [[['last', 'week'], 8.5]],
'where': [(['Syria', 'air', 'strikes', ':', 'US'], 9),
          (['Douma'], 7),
          (['Damascus'], 6),
          (['US'], 6),
          (['Russia'], 5),
          (['Britain',
            ',',
            'France',
            ',',
            'and',
            'the',
            'United',
            'States',
            'of',
            'America'],
           4),
          (['Syria'], 2)],
'who': [([('US', 'NNP'), ('and', 'CC'), ('allies', 'NNS')], 9),
        ([('The', 'DT'),
          ('US', 'NNP'),
          (',', ','),
          ('UK', 'NNP'),
          ('and', 'CC'),
          ('France', 'NNP')],
         8),
        ([('the', 'DT'), ('Pentagon', 'NNP')], 7),
        ([('The', 'DT'), ('strikes', 'NNS')], 7),
        ([('Explosions', 'NNS')], 6),
        ([('Russia', 'NNP'),
          ("'s", 'POS'),
          ('ambassador', 'NN'),
          ('to', 'TO'),
          ('the', 'DT'),
          ('US', 'NNP')],
         5),
```

GIVE ME 5W

- We can think about an hybrid approach with DANDELION API.
- 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,..
- PROBLEM: giveme5w extractor works only in English (can be used as main language at the moment)
- PROBLEM: giveme5w do not present the dative element. but we can easily reconstruct using POS
- HINT: Maybe rearrange the sentences after matching as in meta-info discussed previously

ACCURACY-METAINFO

- 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) combined with level of precision of entities retrieved using dandelion.
- Parameters need to be tuned in an experimental way.

SENTENCE & SCORE

- $P(Sentence\ is\ true) = P(ABCD) = P(D)P(C|D)P(AB|CD) =$
- $= P(D)P(C|D)P(A|CD)P(B|CD) =$
- … we do the calculus for A but it is the same for B …
- $= P(D)P(C|D)\frac{P(A)}{P(CD)}P(CD|A)P(B|CD) =$
- $= P(D)P(C|D)\frac{P(A)}{P(CD)}P(C|A)P(D|A)P(B|CD) =$
- $= P(D)P(C|D)\frac{\cancel{P(A)}}{P(CD)}\frac{P(D)}{\cancel{P(A)}}P(A|D)P(C|A)P(B|CD)$
-
- … being $\frac{P(D)}{P(CD)} \cong 1$ … so being negligible such as P(D) …
- $P(Sentence\ is\ true) = P(C|D)P(A|D)P(C|A)P(B|D)P(C|B)$

A : Who
B : Dative
C : What
D : Where & When (context)

VALIDITY 5W

- According to an algorithm similar to simulated annealing (in reverse) we evaluate a first score of trustiness of a news:

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 (similar to previous)
7. Multiply the total result with T and compare with a threshold to decide if accept or not a news

IMPLEMENTATION

WHERE TO START (USER'S PERSPECTIVE)

From 1L or QUEUE

- If both possible -> decision up to user. USER IS NOT GOING TO CHOSE ALWAYS THE BEST
- 1L -> after consensus 5w, check rule of thumb and send to queue
- QUEUE -> think about LUCKY CASE

START 1L VS QUEUE

START 1L

Start on 1L block, after ”5w consensus” check 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.

START QUEUE

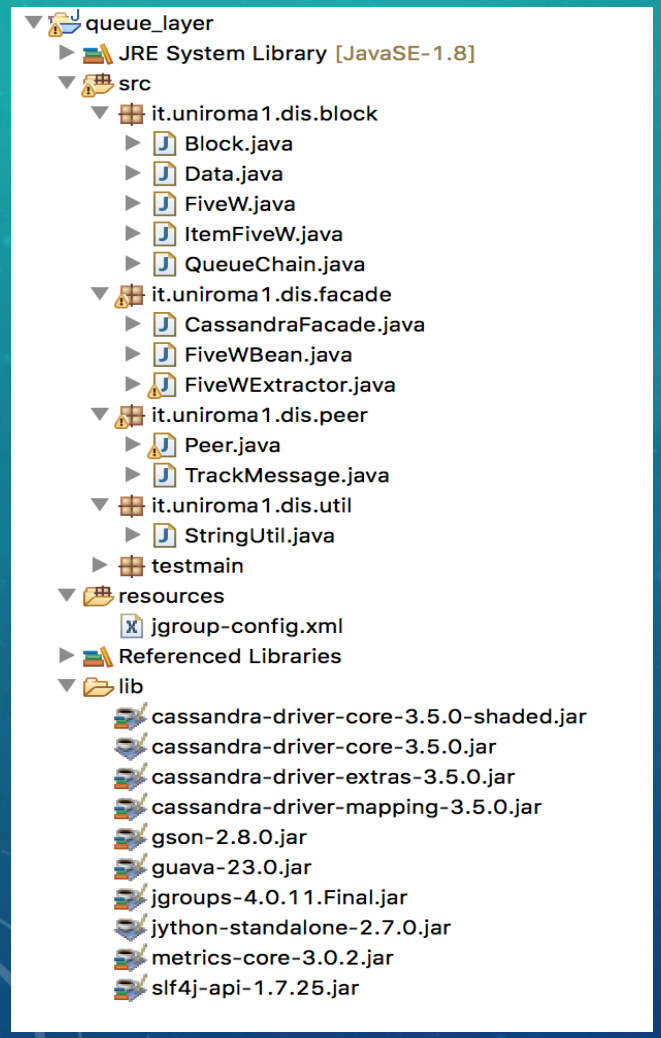
Think about Lucky case.

- User extract 5w and send without trustiness value.
- Consensus on this field (wrt acked sentences & rule of thumb). Peers can ACK or send a NACK concerning to 5w list or trustiness.
- If ACK, save in the queue.
- If NACK, send to 1L and start protocol

LUCKY CASE ZYZZYVA

- HYBRID (IF LESS THAN 3F + 1) SEND TO 1L
  - MODIFIED VERSION, NO USE OF COMMIT CERTIFICATE (JUST FOR PERFORMANCE REASON)
- NEED (PUBLIC-PRIVATE KEY) (IN JAVA HYBRID RSA + AES)
- COMMITED HYSTORY (BLOCK CREATION)
- LEADER ELECTION (ROUND ROBIN) EVERY VIEW CHANGE

QUEUE PROJECT



4 main packages and one for testing purpose

- BLOCK:
  - Blockchain structure and data-mapping for metainfos
- PEER:
  - Peer structure and method to be invoked for implementing the algorithm. TrackMessage is just a message encapsulation.
  - MANAGED BY JGROUP
- UTIL:
  - Common operations such as string to byte
- FAÇADE:
  - It reference the native code used as blackbox.
  - Trigger to python module for 5w extraction, and its ad hoc bean.
  - Reference to Cassandra database.

ALGORITHM (1/2)

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> digest of outcome too*

View managed by Jgroup so we can remove

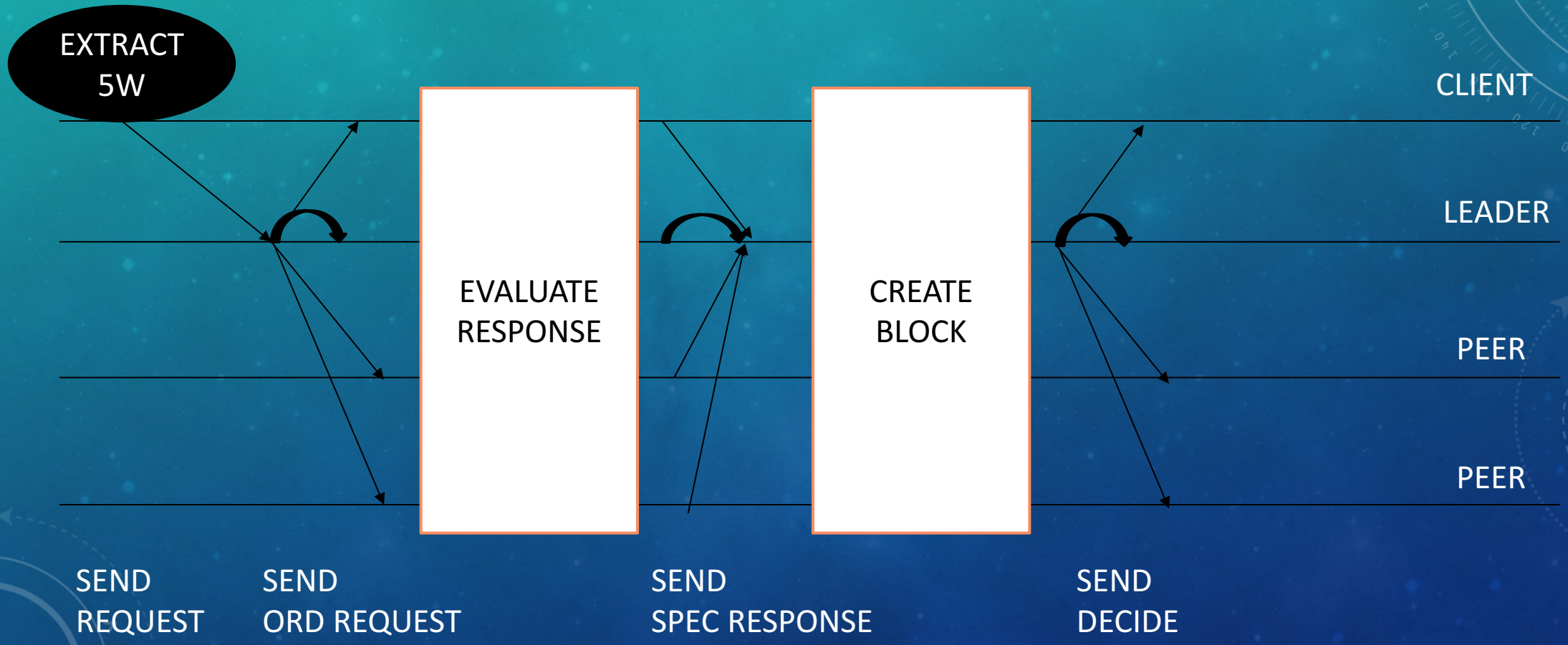
REST OF WORK -SKIP

- *Case A: > 3f .. LEADER adds to its own queue and spread DECIDED (OR OUTCOME IF IS NACK)*
- *Case B: >2f & <3f .. COMMIT CERTIFICATE*

  <COMMIT, leader_pub, CC>σc

  where CC = list of *2f+1* replicas, the replica-signed portions of the *2f+1* matching spec-response messages from those replicas, and the corresponding *2f+1* replica signatures.
- Replica *i* receives a *commit* message from a client containing a commit certificate and acknowledges with a local- commit message *<LOCAL-COMMIT, view_id*, *history_digest*, *message_digest, client_pub, leader_pub >**σi***
- NOW LEADER receives > 2f LOCAL COMMIT OK
- ELSE, SEND TO 1L

SPEED UP QUEUE



BLOCK CHAIN – QUEUE LAYER

- SPEED vs SECURITY
- DATA & BLOCK -> TIMER FOR CREATION, COLLECT AS MUCH AS POSSIBLE

PEER (1)

```
case TrackMessage.REQUEST :
    //YOU ARE THE LEADER IF HERE
    System.out.println("REQUEST RECEIVED from " + msg.getSrc());
    seq_num++; //increase
    Data data = preparData(message.getResource_name(), message.getResurce());
    reply = TrackMessage.getOrderRequest(data, seq_num, hystory, privateKey, publicKey);
    send(reply);
    break;
case TrackMessage.ORDER_REQUEST :
    System.out.println("ORD REQUEST RECEIVED from " + msg.getSrc());
    if (checkAndStore(originalMsg, message)) {
        String outcome = checkValidity(originalMsg.getData());
        reply = TrackMessage.getSpeculativeResponse(originalMsg.getData(), message.getSequence_number(),
                hystory, privateKey, publicKey.toString(), outcome, publicKey);
        sendToLeader(reply);
    } else {
        System.out.println("@@@ NOT CHECK NOT STORE --- " + message.getSequence_number());
    }
    break;
```

PEER (2)

```
case TrackMessage.SPEC_REQUEST :
    System.out.println("SPEC REQUEST RECEIVED  from " + msg.getSrc());
    if (timers.get(message.getSequence_number()) == null) {
        timers.put(message.getSequence_number(),new Timer());
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                try {
                    //assuming it takes 3 secs to complete the task
                    TrackMessage tempMsg = message; //override
                    Thread.sleep(3*1000);
                    if (timers.get(tempMsg.getSequence_number()) != null) {
                        // OTHERWISE TIMER ALREADY STOPPED
                        timers.get(tempMsg.getSequence_number()).cancel();
                        timers.put(tempMsg.getSequence_number(),null);
                        System.out.println("timer***" + tempMsg.getSequence_number());
                        suspect(leaderAddress);

                        //create block
                        decided.put(message.getSequence_number(), ERR);
                        createBlock(message);
                        TrackMessage reply = TrackMessage.getDecidedResponse(message.getSequence_number(),
                                decided, publicKey.toString(), privateKey, publicKey);
                        send(reply);
                    }

                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

        };
        timers.get(message.getSequence_number()).schedule(task, 0);

    }
```

```
//YOU ARE THE LEADER
if (speculativeResponse.get(message.getSequence_number()) == null)
    speculativeResponse.put(message.getSequence_number(), new ArrayList<>());
speculativeResponse.get(message.getSequence_number()).add(originalMsg);
HashMap<String, Integer> temp = new HashMap<>();
temp.put(ACK, 0);
temp.put(NACK, 0);
temp.put(DUP, 0);
for(TrackMessage t: speculativeResponse.get(message.getSequence_number()))
    temp.put(t.getOutcome_response(), temp.get(t.getOutcome_response()) +1);
boolean consensus_reached = false;
String outcome = null;
if (temp.get(ACK) > BYZANTINE_QUORUM) {
    System.out.println("CONSENSUS ACK !!!");
    outcome = ACK;
    consensus_reached = true;
}
else if (temp.get(NACK) > BYZANTINE_QUORUM) {
    System.out.println("CONSENSUS NACK !!!");
    outcome = NACK;
    consensus_reached = true;
}
else if (temp.get(DUP) > BYZANTINE_QUORUM) {
    System.out.println("CONSENSUS DUP !!!");
    outcome = DUP;
    consensus_reached = true;
}
else if (speculativeResponse.get(message.getSequence_number()).size() > CONSENSUS_QUORUM) {
    System.out.println("BYZANTINE MISMATCH");
    outcome = ERR;
    consensus_reached = true;
}
```

PEER (3)

```
        //only leader can insert directly
        if (consensus_reached && !decided.containsKey(message.getSequence_number())) {
            //stop timer
            timers.get(message.getSequence_number()).cancel();
            timers.put(message.getSequence_number(),null);
            //create block
            decided.put(message.getSequence_number(), outcome);
            createBlock(message);
            //LEADER ADDS TO CASSANDRA
            CassandraFacade c = new CassandraFacade();
            byte[] res = StringUtil.fromObjects(hystory.get(message.getSequence_number()).getData().getPayload());
            String hash = hystory.get(message.getSequence_number()).getData().getHash();
            if (outcome.equals(ACK))
                c.insertResource(res, hash);
            reply = TrackMessage.getDecidedResponse(message.getSequence_number(),decided,
                    publicKey.toString(), privateKey, publicKey);
            send(reply);
        }
        break;
    case TrackMessage.DECIDE_REQUEST :
        System.out.println("DECIDE REQUEST RECEIVED  from " + msg.getSrc());
        //leader has just created block
        if (!channel.address().equals(leaderAddress) && !decided.containsKey(message.getSequence_number())
                && checkDecided(decided, message.getDecided())) {
            System.err.println("HERE");
            decided = message.getDecided(); //UPDATE LOCAL
            createBlock(message);
        }
        break;
```

SMART CONTRACT - STRUCTURE

- Initially there were 2 main functions:
  - Add 5w -> the call for inserting a news if above a threshold
  - Evaluate trustiness -> Probabilistic algorithm (should be enriched with SRL and WSD)

- The first function needs to be split in order to extract 5w (python module not easy to integrate)

RETRIEVE 5W

- The Ethereum Virtual Machine cannot access anything but it's own memory space. No calls to the network or to peripherals.

- After extraction I can:
- Make a call to the REST Service from within the smart contract with Oraclize
- Write a service outside the blockchain that periodically checks (polls) a specific state in my blockchain (TX with expected result).

RETRIEVE 5W – FIRST IDEA

- Get some third-party, or a combination of third-parties, to hit the URL for you and tell you what they found. They can do this either by signing the data so that your contract can check the signature (Reality Keys) or by sending the data from their contract to your contract (this is what Oraclize do).
- Take a look at Etheropt at GitHub for a working example.
- The downside of this approach is that your users are required to trust the service that's hitting the URL. If that service is corrupted or hacked, this will result in SFYL. (Oraclize provide a TLS notary proof that the data they provided was the data they supplied you was really supplied by the URL, but that doesn't really help with the actual security risk; Detecting that they gave you the wrong answer is easy, the problem is that your contract is still going to accept their answer even though everybody knows that they're lying...)

RETRIEVE 5W - SOLUTION

- Include the 5w extraction in queue layer, message should be encrypted with leader PRIVATE.
  - PROBLEM : leader hacked
- CONSENSUS 5W … voting mechanism by peers
  - Smart contract has to receive up to QUORUM MATCHING requests
  - EVALUATE QUORUM (F, 2F, 3F +1)

SMART CONTRACT - SOLIDITY

```
function startFiveW(string name,
      string  hash,
      //byte[] payloadRes, //USED AS INPUT FOR EXTRACTING FIVEW
      string claims //WITH "-" AS DELIMITER FOR STRINGS
      //string meta5w //SEPARATED BY CUSTOM DELIMITER, FOR NOW SKIP TILL IMPLEMENTED
      ) public {

    //start saving in temp variable
    Metainfo memory meta;
    meta.name = name;
    meta.hash = hash; //MAYBE COMPUTE
    meta.claim = split(claims);
    meta.state = State.NewlyCreated;

    //**********************
    //USE PAYLOAD FOR EXTRACTING 5W TODO

    //suppose we have as result a string[]

    //consensus 5w TODO
    //**********************

    //compute trustiness
    //ALGORITHM
    uint trustValue = 1;
    //for all string (5w) extracted computeTrustiness, then average, but now do algorithm just one time to test
    //for testing just one
    meta.trustiness = computeTrustiness("Bob","Rome","April", "met", "Alice");
    if (trustValue > 0) {
        meta.state = State.ConsensusTrustiness;
       //add variable to struct
       news.push(meta);
    }
    //for now 50000 is treshold
    //consensus whole meta NO NEED

}
```

```
function add5w(string resHash, string extracted) {
    bool found = false;
    //extracted has to be split and added
    //for all senteces extracted
    FiveWList[] memory list;
    FiveWList memory fivew;
    list[0] = fivew; //i entry

    //after read 5w list check and add TODO

    for (uint i = 0; i < votesRes[resHash].length; i++) {
        if (true) { //compareStrings(votes[votesRes[resHash][i]].exctracted5w,list
            votes[votesRes[resHash][i]].adrs.push(msg.sender);
            found = true;
            break;
        }
    }
    if (!found) {
        Vote v;
        v.adrs.push(msg.sender);
        v.exctracted5w = list;
        votes[resHash.toSlice().concat("hash5w".toSlice())] = v;
        votesRes[resHash].push(resHash.toSlice().concat("hash5w".toSlice()));
    }

    if (votesRes[resHash].length > 10){ //DEFINE A TRESHOLD AND TRIGGER TRUSTINESS AND SO ON
        //TODO
    }
}
```

CURRENT STATUS

- Queue layer -> TEST
- First Layer -> BASIC ALGORITHM OK
    - Has to be enriched with WORD SENSE DISAMBIGUATION & SEMANTIC ROLE LABELING
- First Layer -> Extract and agree upon 5w TODO
- Last Layer -> Seen as blackbox, we do not implement how to finally validate news
    - WE HAVE TO DEFINE ARCHITECTURE
- DataBase -> integrated in queue. Further TEST

EUROPE IS GOING TO FIGHT !!!

Il recente comunicato fa seguito ad un report pubblicato a marzo dal EC High-Level Expert Group (HLEG) che richiama ad una maggiore trasparenza delle piattaforme online per combattere la diffusione di informazioni false online. Il prossimo passo della Commissione sarà lo sviluppo del EU-wide Code of Practice on Disinformation che dovrebbe essere pubblicato entro luglio 2018.

Lo sviluppo della tecnologia blockchain sarà anche incluso nelle attività di ricerca del Horizon 2020 Work Programme, considerato "il più grande programma di finanziamento della ricerca e dell'innovazione europeo".

L'11 aprile, la CE ha annunciato la firma di un accordo volto a creare una European Blockchain Partnership sottoscritto da 22 paesi.

Il vicepresidente della CE Andrus Ansip ha già in passato incoraggiato l'EU a prendere posizione sugli sviluppi della tecnologia blockchain, nello sforzo di rendere l'Europa leader mondiale dell'innovazione digitale.

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://github.com/glowkeeper/Provenator
- https://github.com/professormarek/traceability
- https://dandelion.eu/semantic-text/
- https://github.com/fhamborg/Giveme5W.git