Duniter technical description

Release 0.0.2

Elois, Cgeek, Inso, Matograine

Nov 24, 2020

Contents:

Abstract

Many currency principles involve non-equal rights to monetary creation between humans. We propose a monetary creation based on the Relative Theory of Money, which guarantee equal monetary creation for each willing human. This type of currency can be centralised, however, this could lead to censorship and arbitrary choices of the central institution. Thus, strongly inspired by Bitcoin example, we want the currency to be as decentralised as possible, in the transaction network as in the human identification process. We use a Web of Trust between living humans for identification. This web of trust allows us to impose personalised difficulty for transaction validation, keeping the calculation accessible to low-end hardware and allowing all competent members to secure the currency.

CHAPTER 1

Introduction

Duniter is a software created to manage the Ğ1 (pronounce "june") libre currency. The word "libre" comes from french language, meaning "free" as in "freedom". It refers to the equal right of humans to do whatever they want, whitout creating nuisance to other humans. The currency has been launched on the 8th.march 2017. Libre currency is a concept defined by S.Laborde in the Relative Theory of Money (RTM) that was published in 2010. This theory demonstrates the possibility of an invariant monetary unit: the Universal Dividend. Doing so, the RTM answers the question:

> How should a currency be created to match the principle of equality between all humans, now and between generations?

The results of this demonstration implies a monetary creation:

- on a regular basis
- for each human being
- which amount has to be reassessed on fixed intervals according to a fixed formula.

Thus, Duniter project will associate a human to a digital identity. It will use a Web of Trust with specific rules. As the number of members may evolve, the Universal Dividend has to be created according to the formula:

$$UD(t+1) = UD(t) + c^2 \times \frac{M(t)}{N(t)} \tag{1.1}$$

Duniter is based on a decentralized Blockchain. This technical choice allows irreversibility of transaction and uncensorability of trades and identities. While inspired by Bitcoin, Duniter uses a Web of Trust and the Proof of Work to secure the computation network, thus making obsolete the power race model used in Bitcoin.

In this document, we present how Duniter works and the choices we made. The informations are relevant for the v13 of the Duniter Protocol.

CHAPTER 2

State of the art: Bitcoin case

Duniter uses the crypto-currency concept introduced by Bitcoin[1], which is to use cryptographic tools such as signatures to create trustless digital currencies. Duniter fits this definition, but it has two completely different principles than Bitcoin: the Web of Trust and the Universal Dividend. These differences are on both monetary and technical aspects.

2.1 Monetary creation of Bitcoin: a space-time asymmetry

Space-time asymmetry refers to the relative access of individuals to newly created money[2]. Concretely, most existing currencies (c. 2020) are both spatially and temporally asymmetrical for their users. Let's take Bitcoin as an example to understand why.

2.1.1 Spatial asymmetry

When new Bitcoins are created, only some Bitcoin users (the miners) are given new Bitcoins, while everyone else get nothing. We believe this is the first injustice. However, some might say:

> "Miners used their electricity and time to get it!"

We answer that this work should not be rewarded by newly created Bitcoins. New units should be distributed to the whole community. Miners should be rewared another way, but not by money issuance. Of course, Bitcoin cannot create money through Basic Income since Bitcoin users are not strongly identified, and one might benefit from money creation multiple times if they owned several wallets. Duniter gets rid of this problem by identifying its users and creating the same amount of Basic Income to everyone.

[1] Bitcoin Whitepaper, S.Nakamoto, 2008: bitcoin.org/bitcoin.pdf
[2] Relative Theory of Money, S.Laborde, 2010: en.trm.creationmonetaire.info/

2.1.2 Temporal-asymmetry

Bitcoin has an absolute limit of 21 million BTC (its unit of currency), which means ever fewer bitcoins will be created over time until no new BTC are being generated. So, once the first adopters have mined every bitcoin, how will future joiners get Bitcoins? Just like most of us do for Euros or Dollars: to get money, they will have to work for the ones who already own it.

We believe this is the second injustice. Every member of a monetary community should be equal concerning monetary creation, and get the same relative amount of money over time, even if they are a late adopter. Duniter aims to fix this by making the Universal Dividend (a.k.a. UD) grow by the time according to precise rules, thus making members equal toward money issuance on a half-lifespan.

Most currencies present one of these two asymmetries, including metal currencies and mutual credit, as exposed in the RTM.

2.1.3 A solution

Bitcoin has taught us that it is possible to create a currency system allowing one to both create digital money and to exchange it without a central authority. What we need to change is the way money is issued so we finally have a symmetrical system. We need **Bitcoin + Universal Dividend**. But Universal Dividend implies that the community consists of only identified people. This is where the Web of Trust (WoT) comes into place.

This concept, introduced by cryptography with the OpenPGP format[3], allows us to identify people in a decentralized manner. It works as follows: each person creates a personal identity that is linked to its cyptographic certificate. The identity must be confirmed by others members who use their own cryptographic key. It is that simple: people choose who is part of the community and who is not, not a central authority.

However, Duniter will not use OpenPGP for its cryptographic features: Elliptic Curves[4] will be used instead for the conciseness of its generated keys and its pratical advantages. Duniter has its own Web of Trust principles, that will be exposed later.

2.2 Proof-of-Work mining: a power race

In Bitcoin Model, the calculation and incentive principles cause a power race: new Bitcoins are created for the owners of the most numerous, powerful (and energy-consuming) computers. This leads to a power race an places the control over the currency in the hands of the richest hardware owners. We want to make Duniter blockchain validation much less energy and hardware consuming while keeping a strong level of security. This will be further explained later. A consequence of this choice is the participation of low-end hardware in the Duniter network, leading to a better decentralization of blockchain validation.

2.2.1 What about Proof of Stake?

Proof of stake consensus algorythm was first introduced in 2012[5]. The basic principle is to allow the richest wallets to issue blocks, putting their coin balance as a "stake" they would lose in case of cheat.

At the time of conceiving Duniter, the PoS algorythms had not been tested enough to be used as a fundamental base. We did not chose this consensus principle. Moreover, the principle of allowing owners of large amounts of money to validate transaction can only lead to placing power over the currency in the richests hands: this is contrary to the symmetical principles of a libre currency.

[3] OpenPGP protocol defines standard formats for encrypted messages, signatures, private keys, and certificates for exchanging public keys. The GNU Privacy Handbook, M.Ashley, 1999: www.gnupg.org/gph/en/manual.html#AEN335

[4] High-speed high-security signatures, D.J.Bernstein, N.Duif, T.Lange, P.Schwabe, B-Y.Yang. Journal of Cryptographic Engineering 2 (2012), 77–89. ed25519.cr.yp.to/ed25519-20110926.pdf.

[5] PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, S.King & S.Nadal, 2012: archive.org/details/PPCoinPaper

CHAPTER 3

Duniters Blockchain

Duniters Blockchain follows the basic principles of Bitcoins. This is essential for synchronization between peers, as to prevent double-spend attacks. However, Duniters Blockchain will store different informations than Bitcoins.

The basic use of Blockchain will be registering transactions. For this part, we use the same principles as Bitcoin: transactions have inputs (spending accounts) and outputs (receiving accounts). But contrary to Bitcoin, no generation transaction exists: monetary creation happens only through UD. So, in Duniters Blockchain, Inputs can be either:

- a former transaction (as in Bitcoin)
- a Universal Dividend (specific to Duniter).

Duniters Web of Trust is also written in the Blockchain. The identity of each member gets registered much like transactions are, with a strong link to the time reference. Thus, the Blockchain is a representation of a space-time frame of reference, where "space" are members of the WoT and "time" the basic blockchain units: the blocks. On each point of time, one can determine which account is legitimate to create the UD, only with a blockchain analysis.

3.1 Spam countermeasures

An issue of most cryptocurrency projects is to prevent the common ledger from growing too much. This would require nodes to have a lot of storage and computing power to be usable. In particular, we don't want an attacker to be able to make the Blockchain grow too fast. Most projects implement transaction fees as a way to prevent this, making the attacker lose money. We don't want to introduce this mean: a currency with automatic fees on transactions is no more neutral since it creates an incentive not to spend the money. Several countermeasures against such spam attacks are implemented.

3.1.1 Minimum output amount

Fixing a minimal output amount reduces the power of an attack. Duniter deals with cents of Ğ1 or 1/1000 of the first UD. An attacker could create thousand accounts with only 1 UD. To prevent this, a valid transaction must have output amounts of minimum 1Ğ1. This reduces the power of an attack by 100.

3.1.2 Limited block size and chainability

The block size is always limited. While the protocol allows this limit to evolve to address scaling issues, an attacker cannot register as many transaction as they wish.

With the same goal to prevent too many transactions to get registered, while transactions can be "chained" (refer to another transaction in the same block), the chainability of transactions is limited to 5.

3.2 Scaling

Most of the time, the scaling issue rises for distributed systems that should work on a very large scale. This is not the case of Duniter, for multiple reasons:

- Ğ1 is the first libre currency, and is still experimental on the monetary creation principle. We don't want it to reach the whole world, we only want it to work, to validate or invalidate the RTM. Moreover, the rules chosen for the Ğ1 WoT should limit its size to around 16 million members.
- We foresee the creation of multiple libre currencies aside to Duniter/Ğ,, that would fit local or regional economies. As a consequence, Duniter would deal with less transactions than if it was a world-scale system. The RTM proposes a formula to calculate the exchange rate between two currencies, that could be used to create automatic exchanges for a member travelling away from their community.

However, Duniter has assets that will help if the number of users and transactions grow.

3.2.1 Dynamic block size

While Bitcoin has a fixed block size, Duniters blocks size can evolve. On low use of the blockchain, the maximal block size is 500 bytes. On high use of the blockchain, the maximal block size would be 110% of the average size of the current window blocks(see "personalised difficulty" part for more information). This way, the blocks are bounded in size, but can slowly grow if a massive and legitimate use of the blockchain needs it. The block size (in bytes) is limited as so:

$$\text{blockSize} < \max\left(500; \text{CEIL}(1.10 \times (\text{average block size}))\right) \tag{3.1}$$

3.2.2 Lightning Networks

The Lightning Networks[6] allow almost instant and off-chain transactions. They were first implemented on Litecoin, and are now on Bitcoin. One of their benefits is to make the blockchain store a lot of transactions at once, thus reducing the growth of the blockchain. The Duniter protocol allows XHX() and CSV() unlock conditions that are necessary to implement Lightning Networks. While not available yet, this payment channel might get implemented when needed.

[6] The Bitcoin Lightning Network, J.Poon & T.Dryja, 2016: lightning.network/lightning-network-paper.pdf

3.2.3 Unit base

As the Universal Dividend grows exponentially, with time Duniter nodes would have had to deal with always largest amounts, eventually reaching the BIGINT limit (SQL databases don't deal with numbers larger than 2^{63}). To avoid this, the amounts are expressed with a unit base in base 10. We want the UD amount to always fit in 4 digits. To manage it, the `unitbase` is updated each time the UD value reaches 100.00: it goes from $99.99 * 10^{(unitbase)}$ to $10.00 * 10^{(unitbase+1)}$. All the unit amounts are thus divided by 10. While this might seem strange, this process has already hapened in state currencies. Moreover, the amounts expressed in UD will not change.

With a monetary growth of 10% each year and a stable population, such a change of unit base would happen each 25 years.

CHAPTER 4

Duniter Web of Trust

4.1 Basic Principles

In order to identify "members" accounts - which create monetary units - and other accounts, Duniter uses a Web of Trust. This can be summarized into few principles:

- Each account becomes a member if it received a minimal number of certifications - 5 for Ğ1 currency.
- Only members accounts can send certifications. Certifications have a limited lifespan.
- A certification indicates that the sender accepts the receiver as a legitimate identity.

The aim of the WoT is to identify a blockchain account to a living human. According to Lauterbach et al.[7], the strengh of a relationship should be considered when building a vouch system. For this reason, the Ğ1 Web of Trust rules are expressed in a licence stating what WoT certifications are. A certification represents a strong human relationship: one may certify a close relative, not an acquaintance. Each member has to accept this licence before being included in the WoT. Thus, if a member is part of an attack on the currency, they can be found by mutual relatives. Additional security rules occur to prevent cheat and attacks on a large scale.

Note that non-members accounts can use the currency, but cannot create money. Non-members accounts can be used by individuals as secondary wallets, or by institutions.

We were inspired by the OpenPGP Trust system[8]. However, the OpenPGP trust principles aim at defining trust relatively to a particular user point of view while Duniter needs to identify all valid identities. To achieve these two different goals, OpenPGP allows each user to tweak its trust parameters individually, while Duniter sets rules for the whole community in the block #0, and don't let users chose their particular parameters.

[7] Surfing a Web of Trust, Reputation and Reciprocity on CouchSurfing.com, D.Lauterbach, H.Truong, T.Shah, L.Adamic: snap.stanford.edu/class/cs224w-readings/lauterbach09trust.pdf

[8] Public key validation on GnuPG manual, M.Ashley, 1999: www.gnupg.org/gph/en/manual.html#AEN335

4.2 Why do we need a Web of Trust?

There are two reasons we need it:

1. To make sure that only one Universal Dividend is produced per member at each specified creation interval. In the Ğ1's case this interval is set as daily `86 400` seconds, it is the *monetary parameter* known as `dt`.
2. To identify the nodes hashing the blocks and assign each a personalised difficulty. This custom difficulty proof of work is there to avoid the blockchain's validation mechanism becoming too centralised as is the case with many 'non-libre' cryptocurrencies.

Monetary parameter: Dunters blockchain has parameters, defined in block zero - the so-called genesis block - that define the behavior of both currency and WoT. At the time of writing the Whitepaper, the Duniter Blockchain Protocol (DUBP) has a total of 21 parameters of which 10 are for the WoT alone. We'll focus on these 10.

Suffice to say that the DU is created every 24 hours - 86 400 seconds. This interval is set through the time derivative `dt` parameter and can have a different value in other implementations of the protocol.

We want to make sure that each member can only have one account. As we all know, achieving zero-risk isn't possible[9]. Our goal is therefore not to create a WoT within which fraud would be absolutely impossible, but instead to discourage it. Here is a rewording of our goal in 4 smaller ones:

1. Make the certification process lengthy enough that all members exercise due diligence and are wary of risks.
2. Make fraudulent acts as hard as we can to the extent that they become pointless.
3. Ensure that any Sybil attacks have a negligible impact on the currency by ensuring that illegitimate double Universal Dividends have no significant bearing on the legitimate monetary mass
4. Slow the growth of 'Sybil regions' to give enough time for the community to react and isolate the threat.

Sybil attack: A Sybil attack is an attack perpetrated on a reputation system through the creation of fake identities. A Web of Trust is a specific instance of a Reputation System.

There are plenty of Sybil attack scenarios we can think of and just as many reasons why their perpetrators would want to carry them out. Our objective is that the configuration of the WoT protects both users and its IT backbone infrastructure against these attacks.

This means that micro-attacks performed by small groups of individuals looking for personal enrichment are of no interest to us. The web's role isn't to deter these attacks, this being instead the role of the community. Just like the town you live in is responsible for providing your tap water and electricity but isn't responsible for any burglaries, etc. Much in the same way, Duniter's WoT guarantees us all a functional currency, but do not detect small fraud.

4.3 The importance of having our own certification system

Centralized identification systems can achieve the goal we want. State Identification is an example. However, this has many drawbacks:

- The authority may have arbitrary criteria for identification, for example preventing people without an official state-provided identity or homeless people to be included in the WoT.
- Payment might be required to get identified, thus making the monetary creation not "free".
- The authority is a point of failure for any attacker.

It is of the utmost importance that we remain free from any state or corporation. The WoT is an answer to this criterion. To this day we depend only on the Internet and yet, were it to fail, there are already alternatives being tested around the world for a decentralised communication network.

[9] The Sibyl Attack, J.R.Douceur: www.microsoft.com/en-us/research/wp-content/uploads/2002/01/IPTPS2002.pdf

4.4 A few foundational concepts on graph theory: a bit of vocabulary

- **Graph**: set of points - called 'vertices' - joined by edges - called paths/walks.
- **Simple graph**: a graph with no loops and with no multiple edges. That is, each edge connects two distinct endpoints and no two edges have the same endpoints. A simple edge is an edge that is not part of a multiple adjacency - of edges -. In many cases, graphs are assumed to be simple unless specified otherwise.
- **Directed graph**: a graph in which the edges have a distinguished direction, from one vertex to another. A directed edge can also be called a path or walk. Arrow A –> B is therefore different from arrow B –> A.
- **Endpoints**: the edge with vertex A –> B has A and B as endpoints, respectively as start and end of the path/walk.
- **Isolated vertex**: a vertex whose degree is zero, that is, a vertex with no incident edges.
- **Degree of a vertex**: number of its incident edges - in and out.
- **Out-degree of vertex A**: number of outbound edges / tail ends from A.
- **In-degree of vertex A**: number of incoming edges / head ends to A.

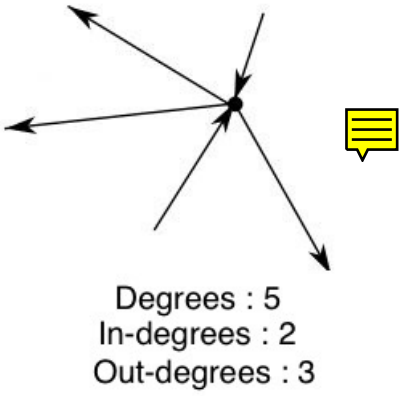

Fig. 1: degrees of a vertex diagram

- **Path**: - aka "walk" - path to follow to get from vertex A to vertex B.

4.5 Definition of the Duniter Web of Trust

The Duniter WoT is a simple directed graphs without isolated vertices. The vertices are the members and the edges are the certifications given and received.

Directed means that the responsibility of issuing a certification is unique and personal to the certifier. The trust they place in the receiver cannot be imposed in the other direction although in most circumstances both parties equally trust each other.

In addition, all vertices are either currently active members or past-members. Past-member vertices are in a specific 'deactivated state' and can no longer issue or receive certifications although the ones already issued or received to/from other members are still considered 'pending' to avoid a collapse of the WoT. If these old members don't come back into the WoT, their pending certifications will eventually expire and they will switch from 'deactivated' to 'isolated' vertices.

To wrap up with old members, after a certain period of time - set in the currency's parameters - their deactivated vertex is removed from the web and the associated identity is 'revoked'. The person who owned the account can no longer use this identity but is free to join the web with another one.

Identity: An identity is a set of three pieces of information: a public key, a name and a blockstamp. A blockstamp points to a specific block in the chain. Its use is to link the name to the public key and freeze the point in time at which an identity was created.

An identity can be in any one of 5 different status: pending, member, old member, revoked or excluded.

Let's take a simple example:

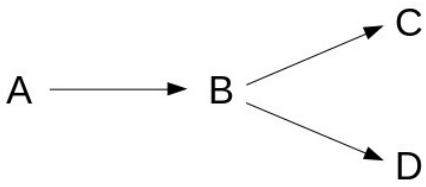

Fig. 2: Simple web of trust

If, for whatever reason, A were to lose its member status, the web would crumble and all other members would be excluded as a consequence. To avoid this, the certification from A –> B will remain valid until its expiry date, leaving enough time for B to receive certifications from C or D.

Because our WoT doesn't have any isolated vertices, each new identity created needs to be pulled into the web with all of the certifications it has received, all in the same block. This calls for a temporary 'buffer' storage space for **pending** identities and the certifications they have received. This storage space is called 'the pool' of Duniter nodes, which we could also have called the 'sandbox'. Duniter nodes include other 'pools' for other documents and metadata not mentioned here.

4.6 Exploring the rules behind a Duniter Web of Trust

The Duniter WoTs work with a set of eight fundamental rules enforced through eleven different parameters. Ten of these parameters are set within the genesis block, the eleventh one - `msPeriod`- having being hard-coded in Duniters code subsequently.

4.6.1 Distance rule and referent members (`stepMax` and `xPercent`)

These two parameters are closely linked and together define the 'distance rule'. The 'distance rule' can only be described after defining what a 'referent member' is:

Referent member: Let D be the total of degrees of a member, and N the number of members. A member is said to be 'referent' if and only if:

$$D \geq CEIL(\frac{N^{-1}}{stepmax}) \tag{4.1}$$

As the size of the web will grow this number will grow too, meaning it will take more certification issuances to become a referent member. The number of certifications needed to become a member shouldn't change. Referent members have a highest degree of centrality in the WoT, compared to other members.

Let us now define the distance rule:

Distance rule: A member is said to observe this rule if and only if for a subset `xPercent` % of referent members R there exists a path of length less than or equal to `stepMax` between R and A. In other words, `stepMax` is the maximum distance with which member A must join `xpercent` % of referent members to become or stay a member.

Referent members only exist so that the distance rule can take effect, they have no special privileges over non-referent members. In a perfect web, that is one in which each member has certified all members they legitimately can, all members would be referent members. However, because the web progressively grows in size and because

members die and are replaced by new ones, there are always members at any given time `t` who haven't yet certified all members they legitimately could. These members would hinder the evolution of the web if they were taken into account in the calculation of the distance rule and the web would effectively stop growing.

Because verifying the application of the distance rule is calculation-greedy, it is only performed when a new identity gets confirmed into the web or an existing member gets renewed. There is an exception to this rule: the distance rule is not observed in the genesis block - when the web is first implemented.

4.6.2 Rule of the minimum number of certifications needed (`sigQty`)

This is the simplest rule, it essentially says that each member must at any given time - meaning in any single block - have received at least `sigQty` active certifications. If, for whatever reason, member A were to have less than `sigQty` active certifications in a given block, they would cease to be a member and be required to publish a request for identity renewal.

4.6.3 The membership renewal rule (`msValidity`, `msPeriod` and `msWindow`)

Bear in mind that a membership doesn't last a lifetime but instead has a lifespan set to `msValidity` seconds.

Every single member - or old member who hasn't revoked his identity or been excluded - can request a membership renewal so long as the last request was made more than `msPeriod` seconds ago. If a member has never requested a renewal, the date of last renewal is equal to the timestamp at which his membership was first created. A new request will be stored in the 'pool' for a maximum of `msWindow` seconds before it's included in the blockchain. Once again, this can only happen once/if the member meets both the `siqQty` rule and the distance rule - if these criterion are already matched it's just a case of waiting for a new block to be minted.

If a member hasn't requested a renewal for longer than `msValidity` seconds, they automatically cease to be a member. From this moment on, the ex-member has another `msValidity` window to renew their membership. When this period of `2 × msValidity` runs out, the membership will expire and this identity will never be available for use again in the web. If the person so desires, they will have to publish new identity and membership documents and find enough certifiers, as any newcomer.

4.6.4 Rule of certification lifespan (`sigValidity`)

All certifications included in the blockchain expire **sigValidity** seconds after they were **issued**.

> /!\ The issuance and the inclusion of a certification in the blockchain occur at different times. When member A issues a certification at time t1, it gets stored in the pool starting at t1 and only finds its way into the blockchain at t2 when all of the web's rules are observed. Several weeks can thus go by between t1 and t2!

4.6.5 Rule of limited supply of active certifications (`sigStock`)

By 'active certifications' we refer to certifications included in the blockchain and that haven't yet expired.

The total of active certifications issued by any member at any single time must be less than or equal to `sigStock`. When this threshold is reached the member will have to wait for one of his active certifications to expire before he/she can issue a new one.

4.6.6 Rule of the time period between two certification issuances. (`sigPeriod`)

As soon as a certification issued by member A gets included in the blockchain, they will be unable to issue a new one before another `sigPeriod` seconds.

4.6.7 Expiry of a certification issuance (`sigWindow`)

When a certification is issued by member A, it will be stored in the 'pool' for a maximum of `sigWindow` seconds. If the certification hasn't been included in the blockchain by then, it will be cancelled and the member's `sigStock` will be repleted by one.

4.6.8 Lifespan of a 'pending' identity (`idtyWindow`)

When a new identity is created, it is stored in the 'pool' for a maximum of `idtyWindow` seconds. If the person hasn't achieved member status by then, the certification will simply be cancelled.

4.7 Details on some of the WoT's peculiarities at the genesis block

The aforementioned rules can only be enforced with an existing web. They cannot be observed when first creating the web, that is when defining the genesis block.

Only rules 2 and 5 - `sigQty` and `sigStock` - can be observed at the genesis block.

The genesis block has to be manually created by the founding members. In practice this means that there must be a choice on which identities to include on the premise that all of them observe rules 2 and 5. In addition, the genesis block must be signed with the private key of one of these identities.

As soon as the genesis block has been created, the other identities can start mining the blockchain and the member who created block #0 effectively looses the decision power they had at creation.

4.8 Why these rules and application cases in the Ğ1

4.8.1 Distance and maximum size

The distance rule is there to curb the maximum size of a Sybil region as well as that of the monetary community as a whole. The `xpercent` parameter prevents the creation of a 'faction' that could take hold of the blockchain.

The Sybil regions are isolated from the rest of the graph in the sense that they can only receive certifications from other ill-intentioned Sybil members. As a consequence, the shortest edge/path between a legitimate member and a Sybil one has to have the attack's author as an endpoint. The maximum depth the Sybil region can attain is therefore contingent on the distance between the attacking edges and the `xpercent` closest referent members, this distance is known as `stepAttackers`. The maximum size of a Sybil region created by `sigQty` members depends on the L parameter, defined as `L = sigQty/sigStock`:

$$MaxSybilSize = (sigStock - sigQty) \times \frac{1 - L^{(stepMax - stepAttackers)}}{1 - L} \tag{4.2}$$

The maximum size of the Web of Trust is given by the following formula:

$$WoTmax = sigStock \times L^{(stepMax-1)} \tag{4.3}$$

However we know for a fact that members will never use all of their available certifications. According to Dunbar[10], on average, one is able to maintain relationships to around 150 people. Being conservative, we will consider that on average, each person will certify 50 accounts. We can calculate the size of the average web of trust

[10] Neocortex size as a constraint on group size in primates, R.I.M.Dunbar, Journal of Human Evolution, 1992

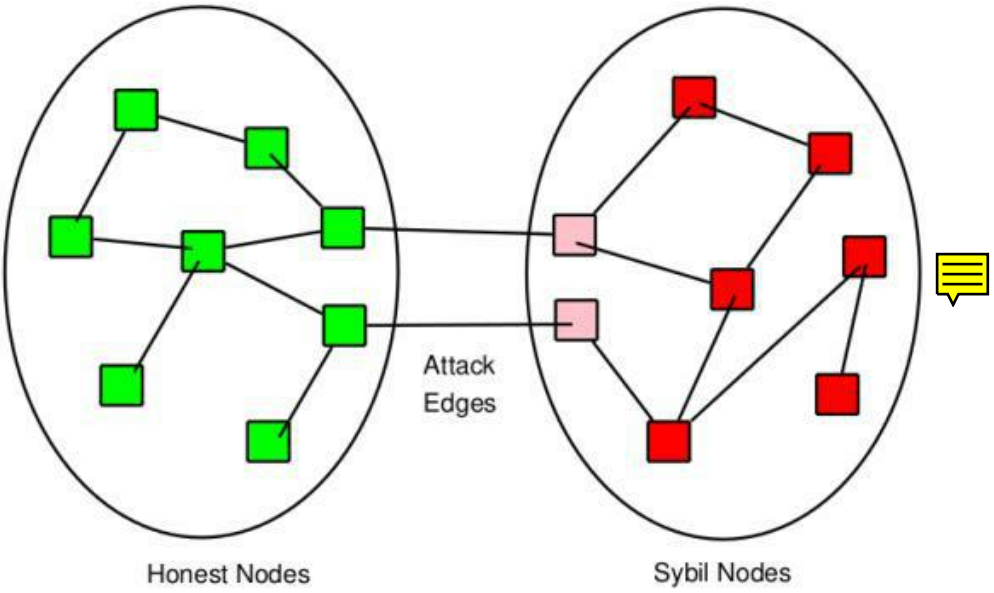

Fig. 3: Sybil region

`WoTavg`:

$$WotAvg = 50 \times \left(\frac{\text{sigQty}}{50}\right)^{stepMax-1} \tag{4.4}$$

Our goal with the Ğ1 is to create a community of about one million members to test the consequences of a libre monetary system. Let's see how we can tweak the pair of `sigQty` and `stepMax` to reach this size:

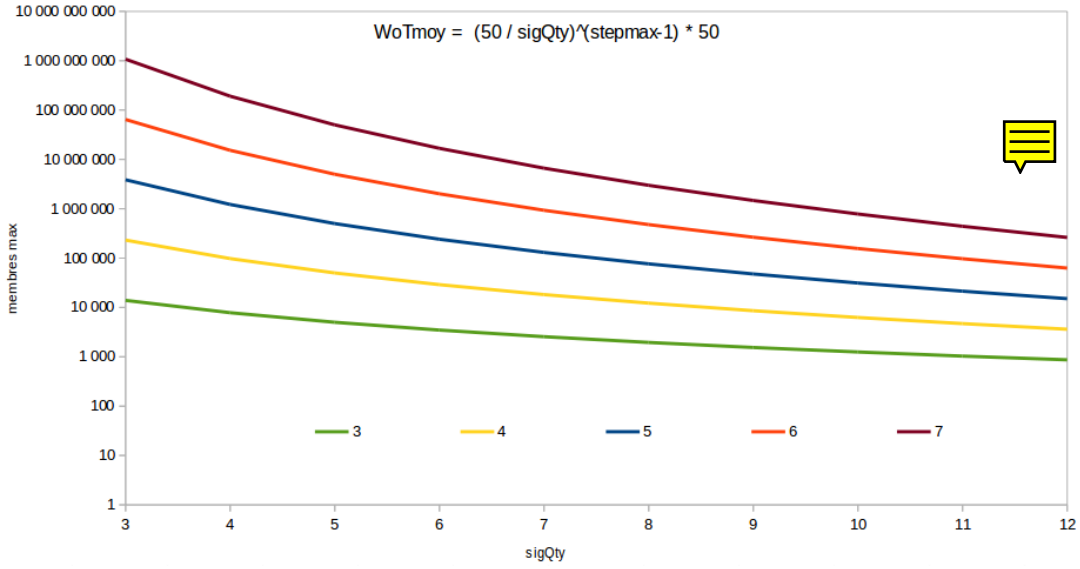

Fig. 4: Average WoT size graph as a function of sigQty and stepMax

The maximum size of a Sybil region grows linearly with `sigQty` but exponentially with `stepMax`. Logic has it that we need to keep `stepMax` as low as possible to ensure sufficient strength to the web. The above graph shows that the lowest value of `stepMax` for a web of a million members is of 5. This is an order of magnitude and is likely to be much higher in reality, we cannot measure it for sure.

For `sigQty` we can choose a value of **4** for a web of **1.5 million members** or **5** for **half a million members**. Bear in mind these are gross figures and could be significantly higher, we are talking anywhere between 1 and 10

million in reality. Calculating WOTavg gives us a pretty good idea of how the web would scale bearing in mind that it considers all members are referent members too - which isn't the case as explained previously -. Hence the maximum size of the web is likely larger, a ballpark figure of half a million is enough for now especially knowing that the smaller `sigQty` is, the easier it is to launch a Sybil attack -it's easier to find four accomplices than five-. For security reasons we have settled on five:

$$stepMax = 5$$

$$sigQty = 5$$

$$sigStock \geq 50$$

The maximum size of a Sybil region therefore is:

$$\frac{(sigStock - sigQty) \times 1 - \left(\frac{sigStock}{5}\right)^{5-stepAttackers}}{1 - \frac{sigStock}{5}} \tag{4.5}$$

with sigStock = 50 we have a Sybil region of:

$$\frac{45 \times 1 - 10^{5-stepAttackers}}{-9} \tag{4.6}$$

A good practice for protecting the web is to maximise `stepAttackers`. That's why we decided that referent members in the genesis block had to be at least four steps away from each other.

Another way to keep a Sybil attack at bay, were it slow enough for members to notice it, would be for referent members to 'stretch' the web intentionally to limit the growth of the region by ensuring that the attackers' legitimate certifications received in the first place aren't renewed. But what if both accounts were created and certified each other super fast and following all rules, how would we counter that? By introducing a minimum length of time between two certifications!

4.8.2 Time is our friend

To help us deter a Sybil attack, we've decided to impose a minimum period of time between any two certifications issued from a single account. This parameter called `sigPeriod` affords us a greater chance to detect the formation of a 'hostile' faction.

Here is a graph showing the evolution of a Sybil region with the variation of `sigPeriod`. The simulation considers that honest members and attackers both issue a certification each `sigPeriod` interval, in days:

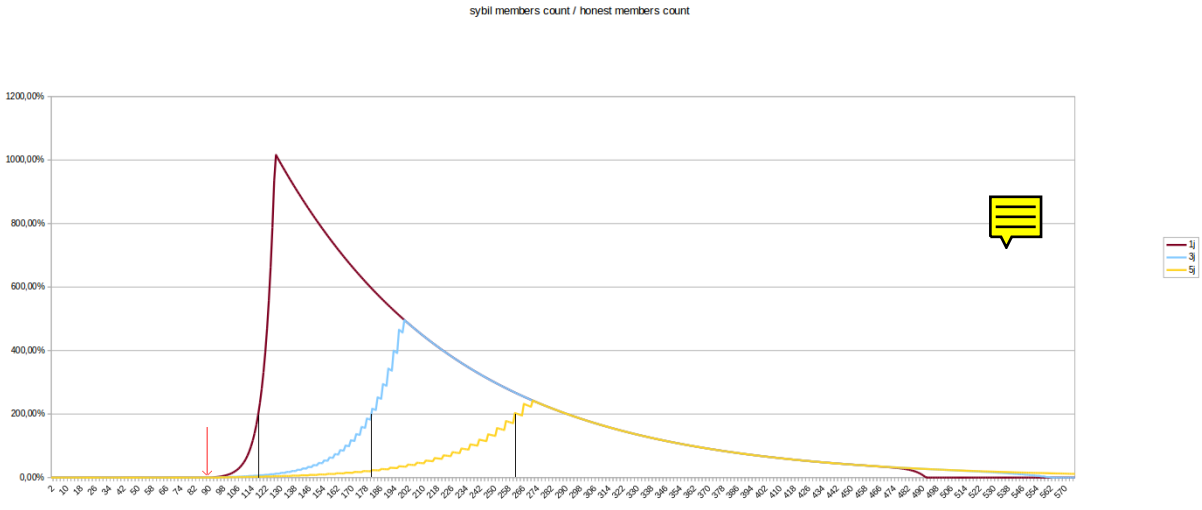

Fig. 5: size of the WoT according to sigPeriod and stepAttackers

As we see, there is a strong link between the growth speed of the region and `sigPeriod`. As evidenced here, we need a `sigPeriod` high enough in order to ensure that the legitimate web can grow at least as fast as a Sybil region. In addition, the higher `sigPeriod` is, the more members will exercise their certification power gingerly, the action coming at a higher 'cost'.

There are numerous advantages to giving `sigPeriod` a high value and no technical barriers to it, hence our choice of five days.

We could have also gone for one week for the sake of simplicity. However there is an underlying idea behind our choice which was quite simply the pace of today's life. Certifying someone can be a lengthy process as one needs to make sure they are correctly applying the Ğ1 licence and people nowadays wait for the weekend to enjoy a bit of free-time. Thus the idea to allow one to certify at the end of every working week -five days- instead of a whole calendar one.

4.8.3 Trust me now, trust me forever? (`sigValidity`, `msValidity`)

There would be two main drawbacks to a lifetime membership in the Ğ1's Web of Trust:

- First of all, some members will die or leave the currency: those accounts should no longer produce the Universal Dividend.
- Secondly it is of the utmost importance that 'rogue' accounts can be excluded from the web at some point.

To achieve this, certifications have a limited lifespan. Members need to seek renewal from their peers after `sigValidity` time. On the other hand, this time can't be too short that members would spend more time seeking renewal than they would exchanging in the currency. Furthermore, a certification with too short a lifespan would foster careless certifying behaviours. The act of certifying must have a high-enough 'perceived' cost to make it feel like an important act. Lastly, we also wanted this lifespan to be easy to remember. Historically speaking, we first settled on the values of `sigPeriod` and `sigStock`, meant one could issue all of their certifications in 495 days, one year was therefore not long enough. We deemed three years to be too much and that's how we agreed on two years in the end.

Thinking that a deceased member could continue producing the UD for two long years without anyone benefitting from it was also something we needed to address. We chose a value of one year for **msValidity**. The act of renewing every year is done through one of the clients interacting with the blockchain, through a simple click on a button. This parameter is less important than others and is mostly there to 'prune' the web of past or inactive members who don't renew their membership.

4.8.4 Keeping the pools free of information glut (`idtyWindow`, `sigWindow`, `msWindow`)

The pools need to be cleaned up on a regular basis to avoid them clogging up with information and to ensure that machines with less calculating power can still run a Duniter node.

To achieve this, identities with pending membership approval and the corresponding certifications have to remain the shortest time possible in the pool while still having a chance of making it into the blockchain.

For the Ğ1, our opinion was that two months would be enough for all potential certifiers to agree on a specific identity to certify. We also wanted a time period that would be easy enough to remember by all. We settled on two months, and gave this value to all three parameters `idtyWindow`, `sigWindow` and `msWindow`.

4.8.5 Avoiding single members from 'knowing too many people' (sigStock)

We considered that on average, each person will certify 50 people. However, we know for a fact that some members will use more than 50 certifications. The maximum social network of one individual is around 150 people[10]. Being conservative, we settled on a maximum certification number `sigstock` of 100. Since `sigStock`'s impact on the size of a Sybil region is fairly limited, we did not investigate further this parameter.

4.8.6 Avoiding locking minorities (xpercent)

It's easy enough to become a referent member, one of the Sybil strategies could therefore be to create a region of referent members. Such a region would grow slower than otherwise but could confer a locking power to its members by using the distance rule. That's why the distance rule cannot be calculated on 100% of the referent members. Hence the introduction of the `xpercent` parameter which defines the percentage of referent members needing to be less than five edges - steps - from each other.

This percentage needs to be low enough to prevent the formation of a locking minority - referent Sybil members being too far from legitimate referent members. On the other hand, it needs to be high enough so as to restrict the maximum size of the Sybil region through the distance rule. The `xpercent` parameter was one of the hardest to define, therefore we might decide to change its value during the Ğ1 experiment.

We were inspired by the Pareto principle[11]: if at least 20% of members give good density to the web, 80% of the referent members will be five or less steps from any other member -referent or not-. The maximum value for `xpercent` is therefore 80%, anything above that and the distance rule could be too restrictive for legitimate use cases. With security our top concern, we chose the maximum value of 80%.

4.8.7 Spam protection with (msPeriod)

This parameter stands out a bit on its own, as it was added after the genesis block. It is there to protect the Duniter P2P infrastructure against 'spam' attacks. We had to think of a strategy against attacks such as high-frequency membership renewal requests - i.e: in every block, every five minutes - or worse still, hundreds of these requests per minute to flood the Duniter nodes. Without such limits, nodes are supposed to address all renewal requests, even in cases where they were last published five minutes ago! The `msPeriod` parameter was given the same value as `idtyWindow`, `sigWindow` and `msWindow`, i.e. two months.

[11] Pareto principle: en.wikipedia.org/wiki/Pareto_principle

CHAPTER 5

Proof of Work with personal difficulty

As each P2P cryptocurrency, Duniter has a way to synchronize its peers. It uses a proof of Work (PoW) to write the Blockchain on a regular basis, much like BitCoin. However, Duniter has a unique asset: the WoT, where each member represents a unique living human.

This difference might seem minimal, but it has an enormous consequence: while Bitcoin uses a race based on computing power only, Duniter creates a validation frame that is no race. It is more like a lottery where each "winning" member is excluded for a certain amount of time. Moreover, those who have more computing power get a handicap, as a way to let other peers win. All this is possible through the WoT, that allows personalised difficulty while PoW is used for synchronization. All the rules of this PoW/WoT mechanism can be verified by reading the blockchain. As a consequence, a peer only needs to have an up-to-date copy of the blockchain to apply the rules. A view of the whole network is not needed.

Another strong difference is that forging peers are not rewarded by the protocol. There is no economical incentive on forging lots of blocs, neither on having a lot of computing power.

One could say that Duniter uses a PoW that needs very low energy consumption compared to BitCoin: an "ecological" PoW?

5.1 Why do we need Proof of Work?

Duniter nodes share a database as part of a p2p environment. The proof of work (PoW) allows machines to synchronize with each other. In Duniter's case, the blockchain is our database, and acts as a ledger keeping a trace of all transactions, status of the WoT and more. How can we let several machines add data (ie: a transaction) at the same time? In addition, how do we settle on how much time has gone by since the blockchain was last updated? Agreement on time is of the utmost importance as we want to create Universal Dividends on a regular basis, and keep track of membership status, both in human time.

Proof-of-work provides a clever solution to both problems:

1. Any machine can write into the blockchain (create a new block) but is only authorised to do so if it has previously solved a mathematical equation that require a certain amount of work. The challenge has to be hard enough to prevent two machines to solve it at the same time, ensuring the unicity of a block's creator.
2. Solving this challenge takes a certain amount of time, which depends on the calculating power of the whole network. This provides a common ground for defining the needed time reference. A block time is set (ie: 1 block = 5 min) and Duniter adapts the challenge difficulty to get an average duration corresponding to this block time.

5.2 Only members can "mine"

One of Duniter's major differences with other PoW-based cryptocurrencies is that only members are allowed to author blocks. Each block is signed with the member's private key, allowing the algorithm to determine a personalised difficulty.

This personalised difficulty eliminates the rat-race for the most sophisticated and powerful mining equipment. Another benefit is the fact that no "supercomputer" can take control of the blockchain. Lastly, Duniter implements a rotation in forging members thanks to this personalized difficulty.

This lightweight PoW is much less energy-consuming than other PoW cryptocurrencies. Members can mine with anything from a raspberry pi to a privacy-first internet cube.

5.3 How does it work?

5.3.1 The hash (aka digest)

Each block is hashed. The hash is considered as a unique "fingerprint" of the block.

Example of a valid hash:

```
00000276902793AA44601A9D43099E7B63DBF9EBB55BCCFD6AE20C729B54C653
```

As you can see this hash starts with five zeros. It took a large number of trials for the node to find it work for someone's computer. Hence the term "proof of work".

5.3.2 The common difficulty

A common difficulty is needed to settle on a yardstick for our time reference. Its role is to make sure the blockchain moves forward at a steady pace - one block every `avgGenTime` seconds, `avgGenTime` being one of the 20 parameters behind the Duniter protocol-.

This difficulty's initial value can be set to any arbitrary value (`70` in Duniter `v1.8.x`) and then acts as a spring, regulating blocktime creation by increasing itself if the creation interval drops under `avgGenTime` and vice-versa.

How is difficulty applied?

The numeric value of difficulty is taken from an array of possible hashes out of all possible hashes. In DUBPv13 the hash of a block is its sha256 hexadecimal hash.

To understand the difficulty, we make a euclidiean division of the difficulty by 16.

Here's an example with a difficulty value of 82:

$$\frac{82}{16} = 5 \Rightarrow remainder = 2. \tag{5.1}$$

The valid hashes are the ones starting with four zeros and with the fifth character less than or equal to 15 - 2 = 13 (D in hexadecimal notation). The valid hashes are then written as starting with: `00000[0-D]`.

Since the block hash will be always the same for a given block, there is no reason for it to start with any specific string. The block content must change until we find a hash matching the correct difficulty. This is the role of the Nonce.

The Nonce

When a member is forging a new block, his computer freezes the block's content and changes the Nonce until the hash matches the required difficulty.

The nonce allows us to mine a new block by finding a hash. The hash value allows us to determine the difficulty level of the proof-of-work performed. Examples of possible Nonce values:

- 10100000112275
- 10300000288743
- 20400000008538

In reality the `Nonce` value follows a pre-determined format akin to `XYY00000000000`. The Nonce's value isn't the number of attempts but rather a value within a set of possible ones. This is how the Nonce is built:

- X is a number assigned to a specific peer. Let's assume that someone has several nodes each with the same private key, this would lead to possible collisions if this person were to mine the same block with different nodes. Each node will therefore have its own unique X to prevent this from happening.
- Y is the number of cores of the processor. The Nonce starting with `107[...]` belongs to a seven cores processor, while `199[...]` could be the proof generated by a 99 cores processor.

The rest of the Nonce, the part that follows after the XYY, is the numerical space for this individual node and is unique to each of the CPU's core. This space is comprised of eleven digits (`00000000000`). For the sake of accuracy, we use the term CPU in the wider sense, it can be understood as a bi-CPU for example. We take into consideration the number of cores for the resulting PoW.

5.4 Personalised difficulty

Earlier in this article, we explained that the personalised difficulty is the new and key concept that sets Duniter apart from other PoW-based cryptocurrencies.

Here is how this personalised difficulty is calculated and assigned:

It is determined by a combination of two different constraints with complimentary roles: the **exclusion factor** and the **handicap**.

Let `powMin` be the common difficulty, `exFact` a member's exclusion factor and `handicap` their handicap. This member's personalised difficulty `diff` is:

$$\text{diff} = \text{powMin} \times \text{exFact} + \text{handicap} \tag{5.2}$$

5.4.1 Understanding `exFact`, the exclusion factor

Members who have never produced blocks or haven't for quite some time are assigned an exclusion factor of `1`. Their personalised difficulty is therefore simply the sum of `powMin + handicap`.

Before reading on, let's precise the role of this exclusion factor. When a member adds a block to the chain, his `exFact` jumps up from one to a very high value, to prevent them from forging other blocks immediately after and taking control of the blockchain.

The exclusion factor will then rapidly return to one. This delay is expressed as a number of blocks. It is calculated as a proportion of the number of members forging. In the Ğ1's case, this proportion is 1/3, meaning that if there are fifteen members currently forging, the member's exclusion factor will drop down to one after five blocks.

What is intended by "the number of members forging"?

We mean the number of members trying to create the next block. In reality, there is no way to precisely know how many members are calculating at any given time, because it is impossible to view the entire network. But we need this information, whithout which assigning a personalised difficulty is impossible. To achieve this, Duniter looks back at the blockchain and assumes that there is as much members forging as those who have found at least one block in the last blocks in the current window, minus the very last one.

Current window

We use the concept of **current window**. The current window is the number of blocks we look back at to determine how many members are forging. Let's see how it works:

- `issuersFrame` is the size of the current window in blocks.
- `issuersCount` the number of members who have calculated at least one block during the current window.

Both `issuersFrame` and `issuersCount` are block fields. When first starting a blockchain, the very first block has an `issuersFrame=1` and an `issuersCount=0`. The genesis block is excluded as there are no members in the current window.

From the second block onwards (block #1) we track the variation of `issuersCount`. The member having mined block #0 enters the current window and in block #1 we will therefore mention `issuersCount=1`.

`issuersFrame` then varies as follows:

- if `issuersCount` increases by N (with a maximum step of N = 1), then `issuersFrame` will increase by one unit over a period of 5N blocks.
- Conversely, if `issuersCount` decreases by Y (with a maximum step of Y = 2 = current window inching forward + loss of one calculating member), then `issuersFrame` will decrease by one unit during 5Y blocks.
- When such events overlap, `issuersFrame` evolves as so:

bloc	event	issuersFrame
T	Babar writes a block and enters issuersCount	160
T+1	Celeste leaves issuersCount	160 +1 = 161
T+2	N/a	161 +1 -1 = 161
T+3/4/5	N/a	161 +1 -1 = 161
T+6	N/a	161 -1 = 160

The calculation can be found under rules BR_G05 and BR_G06 of the DUP protocol.

exFact and the personalised difficulty

We explained that `exFact` spikes immediately after the member has found a block. It decreases then rapidly to `1` after a number of blocks equal to `1/3 * issuersCount`. Let's see precisely how we calculate `exFact`:

- `nbPreviousIssuers` is the value of issuersCount at the last block `N` found by the member.
- `nbBlocksSince` is the number of blocks found by the rest of the network since block `N`.
- `percentRot` is the number of *not excluded* peers we want. It is a monetary parameter, its value is 0.67 for Ğ1 currency.

$$exFact = MAX[1; FLOOR(\frac{percentRot \times nbPreviousIssuers}{1 + nbBlocksSince})] \tag{5.3}$$

The FLOOR is a simple truncate function. For `exFact` to exclude the member, we need:

$$\frac{percentRot \times nbPreviousIssuers}{1 + nbBlocksSince} >= 2 \tag{5.4}$$

We can see that the member is not excluded if `nbBlocksSince` is greater than 1/3 of the calculating members. Take as an example `nbPreviousIssuers = 6` and `nbBlocksSince = 3`:

$$0.67 \times \frac{6}{1+3} = 1.005 \Rightarrow exFact = 1 \tag{5.5}$$

However, if the member computed a block one block ago (`nbBlocksSince = 1`), exFact = 2 and the forging peer is excluded:

$$0.67 \times \frac{6}{1+1} = 2.01 \Rightarrow exFact = 2 \tag{5.6}$$

Moreover if the last block was authored by the said member (`nbBlocksSince = 1`), then:

$$exFact = 0.67 \times nbPreviousIssuers \tag{5.7}$$

ExFact value increases according to the number of members calculating. Thus, if there is enough members calculating, even mining farms would be excluded. We have therefore succeeded in our intent to deter attempts to seize the blockchain and its currency.

However, at any time `t`, the two-thirds of calculating members all have an exclusion factor of `1`, even though they might not all have the same computational power at hand. If the personalised difficulty only took into account the exclusion factor, then only the members with the highest computational power from the remaining third would be able to author new blocks and the other 2/3s would almost always be excluded. Lesser machines wouldn't stand a chance...

5.4.2 The handicap

The handicap is the second parameter of the personalised difficulty. Its main role is to improve the rotation of forging peers. A higher handicap is assigned to members with higher calculating power, so lesser machines can also compute blocks. As a consequence, there is no incentive on forging with powerful computers. Security can be achieved with less computing power than with pure PoW.

The aim is to handicap the half that has authored most blocks (the most powerful half) to favour the other one. So, the handicap formula will use the median number of blocks authored by peers within the current window.

- `nbPersonalBlocksInFrame` is the number of blocks authored by a single member within the current window.
- `medianOfBlocksInFrame` is the median number of blocks written by the calculating members during the same timeframe.

$$handicap = FLOOR(\frac{\ln(MAX(1; \frac{nbPersonalBlocksInFrame+1}{medianOfBlocksInFrame}))}{\ln 1.189}) \tag{5.8}$$

Let's unwrap the formula:

$$\frac{nbPersonalBlocksInFrame + 1}{medianOfBlocksInFrame} \tag{5.9}$$

is simply the ratio between the number of blocks authored by the peer and the median number of blocks. For example, if a peer has authored `9` blocks in the current window and the median is `5`, then the ratio will be `(9+1)/5 = 2`.

- The MAX function allows us to ensure that the handicap has a value at least equal to `1`.
- The Napierian Logarithm of this ratio prevents the handicap from becoming excluding. We want the handicap to level the calculating field so that all peers stand a chance, not to exclude peers.
- If we wanted the handicap to be applied as soon as the median is reached, we would divide it by `ln(1)`. The problem is that we have already set a minimum value of `1` with the MAX function. If we were to divide the ratio by `ln(1)` all calculating peers would have a handicap `\>= 1`. In addition, we chose not to handicap a member who is right on the median.

That's why we went for `1.189` rather than `1`. A member has to be at least `18.9%` above the median to be assigned a handicap. 18.9% is actually `16^(1/16)`, the difficulty factor between two levels of the proof work (hexadecimal hash).

To conclude, you have to remember that:

- The handicap is indexed on the logarithm of the ratio to the median,
- Handicap is only applied on members whose ratio to the median is greater than the ratio between two levels of the proof-of-work's difficulty.

CHAPTER 6

Conclusion

Duniters Blockchain can be compared to Bitcoins: a common document retracing the history of the currency. However, Duniter registers not only trades, but also the history of relationships in the community as a mean to identify a human to a digital account. This way, Duniter has information about the fondamental reference of RTM: living humans. A libre Currency can be issued thanks to the Universal Dividend.

More than that, Duniter proposes a new model for securing a Blockchain in an efficient and decentralized way. Basing the security on a Web of Trust with an individualised security makes the calculation rules more fair. A side effect of this choice is a network consisting mostly of low-end computers, maintaining a good security and helping decentralization of calculation.

The ultimate goal of Duniter project is to allow people to participate in a libre economy, thanks to a libre currency. What is a libre economy? The Relative Theory of Money defines it through four economic liberties:

- The freedom to choose your currency system: because money should not be imposed.
- The freedom to access resources: because we all should have access to economic & monetary resources.
- The freedom to estimate and produce value: because value is a purely relative to each individual.
- The freedom to trade with the money: because we should not be limited by the avaible money supply.

Those 4 economic freedoms should be understood together, not exclusively. Plus, “freedom” has to be understood as “non-nuisance”. So here, freedom does not mean the right to take all of a resource (like water source in a desert) so no more is available to the others. Now you get it, this is the goal: free economy through free currency.