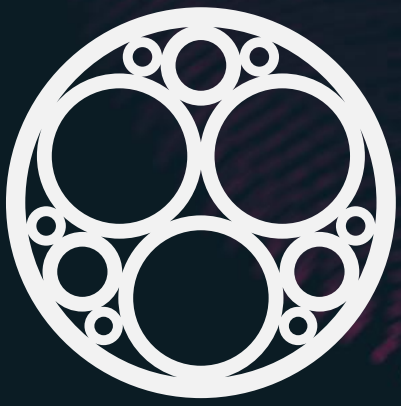


SONM

Supercomputer organized by network mining

sonm.io

SONM

(Supercomputer organized by network mining)

Distributed computing power exchange

Decentralized operating system for fog computing

19.05.2017

sonm.io

Reddit	GitHub	Twitter	Telegram
BitcoinTalk	Slack	Facebook	Medium

TABLE OF CONTENT

1. INTRODUCTION

1.1. What is SONM

SONM is a decentralized worldwide fog supercomputer for general purpose computing from site hosting to scientific calculations. SONM company is an effective way to solve a worldwide problem - creating a multi-purpose decentralized computational power market.

Unlike widespread centralized cloud services, SONM project implements a fog computing[1] structure – a decentralized pool of devices, all of which are connected to the internet (IoT / Internet of Everything).

IoT/IoE, as an important part of the available computational power in the world, is one of the key directions of work for the SONM project. (See further, chapter 2.1)

We use cost-efficient fog computing instead of a costly cloud structure, so there is no more need to pay in advance for private and monopolized cloud computing such as with Amazon, Microsoft, Google Cloud, etc. Moreover, since SONM is fully decentralized, there is no single authority that regulates computing resource distribution.

SONM has a hybrid architecture, and therefore supports any kind of computational task without facing Ethereum's "out of gas" problem.

From a technical point of view, SONM is a top layer of underlying P2P technologies – BTSync for data transfer, Cocaine open source PaaS technology as a decentralized computing platform, and Ethereum Smart Contracts as a consensus system.

There is no central control behind the system and no backdoors or escape hatches. Several existing technologies were combined and modified by our developers to make new SOSNA technology.

In terms of providing distributed value for investors, SONM uses its own token SNM, based on Ethereum's blockchain. (click to read SONM token description in Business Overview).

Almost every online service needs computational power for their product, including websites, online shops, MMORPGs, companies using large databases, and apps. Everyone in the world using the internet for business will have an option to use SONM's tokens in order to solve their computing power issues. Moreover, all internet users will be able to use SONM to receive passive income by providing their computational resources for rent.

This disruptive migration from centralized cloud computing to decentralized fog computing will not happen quickly: it will be a long transition, but the results will be positive. SONM token price calculations show decent ROI for the project's early adopters.

SONM token price is supported by stable market demand for computing power and ability to provide more competitive prices than traditional cloud computing services. SONM token holders earn a percentage from transactions and operations fees (buy-sell-develop). It is a direct analogue of holding shares and receiving dividends from operational profit.

If you are a miner or computational power owner, SONM is a great resource of using your equipment for calculations and processing real tasks.

SONM fog computing platform is a fresh start for solo mining. There are lots of miners with GPU mining farms that have become useless due to the increased Proof-of-work mining difficulty (even for altcoins). In recent years, being a part of a mining pool has been the only way to guarantee profit from mining. But even in the process, this profit is so small that it often does not cover the cost of electricity spent for PoW mining.

SONM platform is the efficient solution for miners. (click to read chapter Goverance in BO)

With SONM you will stop burning your kilowatts for PoW mining and start serving calculations for everyone in the network. For those who are confused by the difficulty bomb or Ethereum (and many others) PoS-migration - each miner is suggested most profitable applications and tasks for their hardware. CPU, GPU, ASIC, and even gaming consoles and smartphones can be used for SONM fog computing. All you need to do is to set up a mining client application and run it.

SONM is a Multi-agent system, so each user will able to use intelligent agents and smart-contracts to maximize profit. You can set your automatization level by choosing each project manually with one-click settings. The SONM system will then automatically pick the most profitable project for your equipment, work with it and receive payouts to your personal Ethereum address.

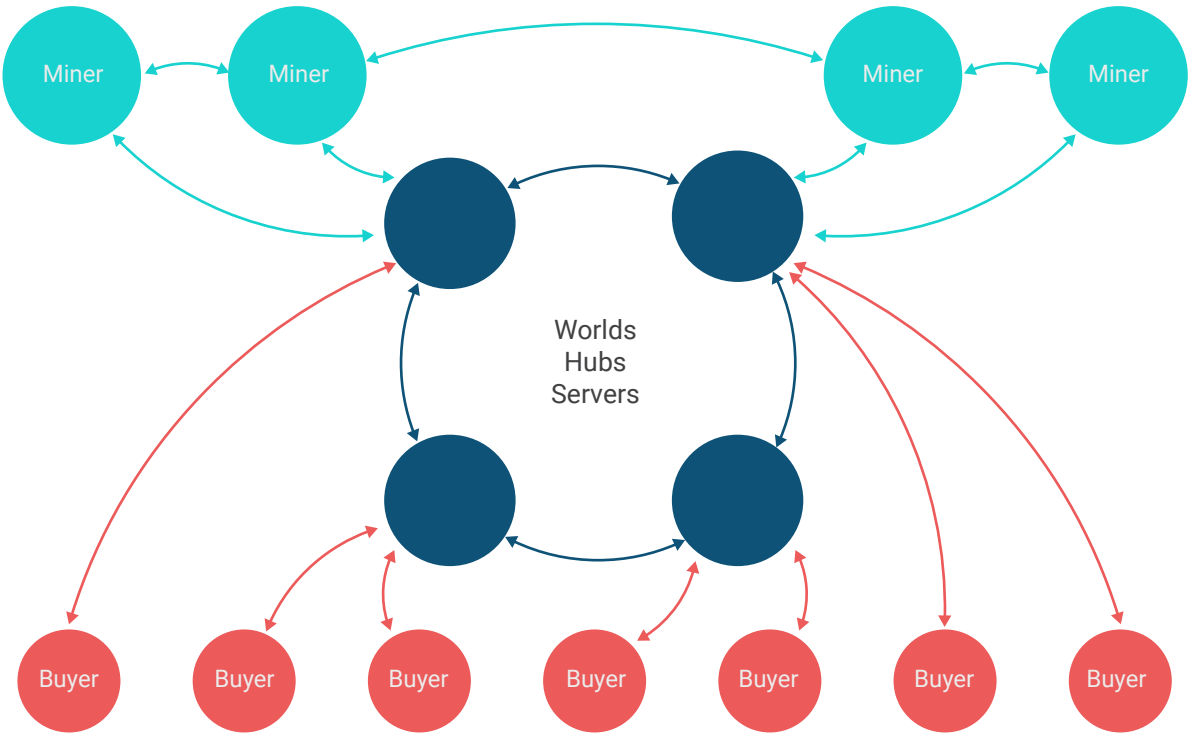
SONM is easy to setup and use, both for miners and computing power buyers. There is no need to have advanced IT skills or to hire an IT specialist if you use SONM — our self-learning system finds the most profitable task for miner's equipment (and vice versa for buyers). The network also runs this task with no need to set up and support a dedicated server.

SONM is Self-learning and totally safe for its users. Our system supports anonymity tools like proxy, VPN or TOR, but it can't be used as a hacker dream toolkit. Intelligent agents are able to self-educate using neural networks and keep malicious users out of the system, while at the same time providing the most efficient task solution - both for miners and computational power buyers.

SONM computing power exchange is the free market, so malicious hubs and users will shortly be ignored by buyers and miners due to their bad reputation. (link to 2.8 Safety and Security)

We expect SONM to be the smartest, cheapest and largest decentralized computing system with strong rules regarding morality and loyalty. This is largely due to SONM's reputation system and self-learning intelligent agents.

SCHEME OF THE NETWORK



1.2. SONM Use Cases

We have experience with the limitations of BOINC itself – it is scientific software and supports only C++/ FORTRAN/Python, therefore it is not flexible. We started using more advanced solutions like Cocaine and Docker container (which support more languages, including Java, Node.js, Go and etc.) We decided that we will go the other way, and will focus more not just on the distributed calculations field like BOINC does, but more on fog computing. This way, we can build a more universal platform not only for scientific calculations. The flexibility of the SONM platform and its multi-purposeness is knit to PoE for non-deterministic task which is a unique technology owned by SONM (Proof of Execution).

1.2.1. Scientific projects

SONM network can be used to run essential scientific calculations requiring massive computing power, for example:

- drug development
- modelling
- meteor trajectory modelling
- bioinformatics
- aerodynamic calculations
- social statistics
- climate predictio

There are several major areas where large computer capacity is needed.

Usually in case of computationally-intensive methods, or (and) tasks with huge input data requirements.

Molecular dynamics and quantum mechanics are examples of such kind of computationally-intensive methods. Molecular dynamics is a computational experiment to study small molecules behaviour and intermolecular interactions. The largest computer power is required to model the interaction between new medicines and biological targets.

Molecular dynamics is also helpful in studying the fermentation mechanisms. Relevant programmes are carried out to model the enzymes behaviour depending on temperature, presence of solvent, pressure and another parameters. The system complexity is proportional to the number of modelled particles and also to complexity of their representation.

For example, the system complexity and estimated time will grow exponentially, along with increasing of protein amount, modelled system volume (the number of water molecules around proteins) and complexity of water models. However, molecular dynamics is based on the using the methods and approaches of molecular mechanics, and does not take into account the quantum effects.

Therefore lets see another example of such methods use. The quantum mechanics and chemistry methods allow now to model just small systems, but give the opportunity to model molecular biological system with maximum accuracy.

Moreover, when medicine development also needs various methods to model the exodus of pre-clinical tests. For example, metabolism modelling, adsorption modeling, toxicity (ADME/Tox) modeling. Such kind of methods are most effective when they achieve chemical compounds in correct operation with large spaces.They should be able to reach hundreds of billions connections in a single database.

The personalized medicine methods development is example of using the project in bioinformatics.

To fulfil such kind of tasks we need to process the huge amount of data for each patient - genomic data (genome is the collection of all patient's genes), proteomic data (proteome is set of all human proteins), transcript data (transcript is full RNA-set), as well as study of their combination and creating dynamic

metabolism models for any human pathology. Nowadays, there are huge efforts to create interactive models of organs and whole the human organism. These models are supposed to model the complex direct impact on the biological targets' set. Selection of such targets and combinations for medicines should be maximally individualized to reach the greatest effectiveness of therapy and to minimize any side effects.

1.2.2. Site hosting

The SONM network can be used to host websites without depending on centralized cloud services (AWS / Azure / Google Cloud etc) or hosting providers. We use Cocaine open source PaaS technology to implement virtual machines recognized as servers, with IPFS and other decentralized data storage solutions as an underlying layer.

Website owners can also use our code snippets on their websites to collect payments in SONM or Ether tokens and automatically pay for hosting, according to market value.

It is important to look into TOR operation (The Onion Router). TOR uses pseudodomains .onion, domain names look like this http://o3shuzjrnpzf2aiq.onion/

Domain names in the .onion domain are generated based on an open random key server and consist of 16 symbols. These websites are actually not websites at all, but are in fact so-called hidden services. SONM is going to implement such services, of which one application could be hosting websites. Storage and operation will be decentralized.

Realization will be in the form of free access to service data from the internet, or similar to the TOR system, limiting the access. The structure of the service depends directly on the application running in the container.

In the address bar the service may look like name,site.sonm or probably just %name%.sonm (which would be the name of the service that finds a hidden node using the locator and loads the website). This can be used for additional identification of services on the SONM network and granting them additional properties.

1.2.3. Game server use-cases

There are lots of MMO games using in-game currencies. Our technology offers a solution for deploying game servers in the **SONM network**. Furthermore, game currencies can be easily exchanged for SONM tokens and back using our out-of-the-box solution.

In addition, gamers can support their favorite game servers by providing their computing resources in exchange for tokens or in-game currency. In example, Quake dedicated server looks like a fully tuned and setted up, ready-to-play, platform-independent docker container.

1.2.4. Neural networks projects

Machine learning algorithms, in particular neural networks, are a powerful technology that has become more and more prevalent in the recent years; machine learning, powered by neural networks that emulate the work of the biological brain, is the true way in which computers can be empowered to act like humans, and sometimes even transcend human abilities. Since the main application areas of ML are sight and speech recognition, translation, prediction and data analysis, it is not surprising that this technology is becoming part of many services. Projects on neural networks require huge computing power for their deployment, training and tuning. The SONM system represents an economical and efficient solution for the implementation of machine learning algorithms and neural networks.

Designing the architecture of deep neural networks and the testing process in retrospect to solve the tasks posed has a number of problems, some of which SONM is ready to solve.

Modern personal computers (for example, Core i5, 8Gb RAM) allow for a comfortable time to train neural networks on samples within tens of thousands of examples, with the dimensions of input data up to several hundreds. Large samples are a task for these deep networks, which are taught on multiprocessor GPUs for the implementation of convolutional and recurrent networks, various activation functions (semilinear, sigmoidal, softmax) and an algorithm for back propagation of the error.

You can use SONM not just to speed up the learning process and improve the learning quality, but also to guarantee quality of the working network, pre-learning the network or algorithm as a predetermined setting.

Using the SONM system you can speed up the algorithm treating processes for pre-computing of data such as:

- task-independent - PCA, LDA, Kernel PCA
- picture treating - SIFT, SURF, CHoG, WAVELETS
- sound treating- DFT, FFT, MEL cepstra
- text treating - ITF-DF, N-grams

1.2.5. Rendering video and computer graphics

Rendering CGI can be distributed over the SONM network between a large number of computing devices and can be processed very quickly (in a matter of minutes).

We provide much faster processing for Buyers' (Clients') CGI computing projects due to SONM's infrastructural flexibility. Compared to one K80 NVIDIA unit rental from Amazon (for example, for 10 hours), a buyer can use the SONM network to rent 600 K80 NVIDIA units with a total task processing time of 10 minutes for each of them. It allows for use of more efficiently distributed architecture and parallel computing.

Unlike cloud computing services, SONM can provide buyers any rental time, any computing architecture and any computing network structure.

1.3. Cost-efficiency for the end-clients

Using the SONM platform will provide beneficial conditions due to a few important factors:

- Lowering the bandwidth costs
- Willingness of miners to use their hardware
- The market will be saturated with new sellers of computational power, which will facilitate a drop in prices
- The lack of centralized servers which require additional infrastructure costs to maintain

2. SONM TECHNOLOGY

Nowadays the popular Internet of Things concept[2] (IoT) gives way to the new emerging concept called Internet of Everything (IoE).

Internet of Everything is the unification of all computing resources of humanity. It has core differences with currently widespread centralized cloud computing technology.

In order to develop a system implementing this disruptive idea, the SONM team used the most efficient and proven P2P, distributed computing and blockchain technologies.

SONM is not a monolith product, it's a top layer built on underlying protocols and technologies: Ethereum, BTSync, Docker, Cocaine, etc.

(By the way, Bitcoin creator(s) also combined existing technologies (cryptography, P2P nodes network, git, Proof-of-work concept, etc) to bring a brand new independent decentralized currency/payment system to the world.)

2.1. IoE, IoT and fog computing

Before describing the future "World Computer" architecture we need to mention some details regarding IoE, IoT and fog computing concepts.

Nowadays, the concept of Internet of Things (IoT) is commonly known.According to the IoT concept, Thing is any natural or artificial object able to have an IP address and transfer data over the network.

Internet of Everything (IoE) represents further development of IoT concept: "Cisco defines the Internet of Everything (IoE) as the networked connection of people, process, data, and things. The benefit of IoE is derived from the compound impact of connecting people, process, data, and things, and the value this increased connectedness creates as "everything" comes online.

IoE is creating unprecedented opportunities for organizations, individuals, communities, and countries to realize dramatically greater value from networked connections among people, process, data, things[3]."

This definition emphasizes a very important aspect of IoE, which distinguishes IoE from IoT: namely, the so-called "network effect", formulated by James Macaulay from the Cisco IBSG consulting department. The term "network effect" refers to a decentralization of organizations included in IoE.

These kinds of decentralized systems are being developed by groups of so-called "crypto-anarchists" (people implementing decentralized P2P systems using cryptographic methods[4]). A way to implement the IoT will be presented in the form of our own operating system, built on CoreOS implementing SONM's functionality. Any device can support such a system and when connected, can act as a computational unit for the SONM fog.

Furthermore, in this document we are referring to decentralized organizations of computing machine resources, and not decentralized human organizations. Most of the data in the current IoT state of development is being processed by private centralized clouds - i.e. using cloud technologies, like AWS, Microsoft Azure, etc.

Centralized cloud technologies have several weaknesses and can't be used in IoE.

Some Things in IoE can create massive amounts of data. Cisco gives the example of the jet engine, which creates about 10 Terabytes of its activity data in 30 minutes.

Transferring this data to the cloud, and receiving the results of data processing, requires adequate network bandwidth, takes significant amounts of time and can have delays.

Furthermore, private centralized cloud systems potentially can be compromised, influenced from the outside, attacked or have failures, and also have lower computing power than fog computing solutions.

> How can these problems be solved?

Fog Computing shifts the cloud computing paradigm and moves it to the lower level of the network. Instead of processing some task using the cloud, we can use all the devices surrounding us: personal computers, smartphones, even coffee makers and traffic lights.

Cisco's Ginny Nichols originally coined the term Fog Computing. The metaphor comes from the fact that fog is a cloud that is close to the ground, and thus fog computing concentrates processing at the edge of the network. In Fog computing, data processing and applications are concentrated in devices at the network edge rather than existing almost entirely in the cloud. That concentration means that data can be processed locally in smart devices rather than being sent to the cloud for processing[5].

Thus, instead of centralized cloud solutions, we can use fog computing systems, getting the computational power of every internet-connected device, with decentralization advantages like independence from any centralized service and full protection against possible failures, etc.

2.2. World Computer

The so-called "computing fog" is the layer of computational resources able to process some kind of task.

However, aside from computing fog, the system also involves its users setting computational tasks, and some middleware distributing these tasks among the fog resources, which then returns the result of the calculations.

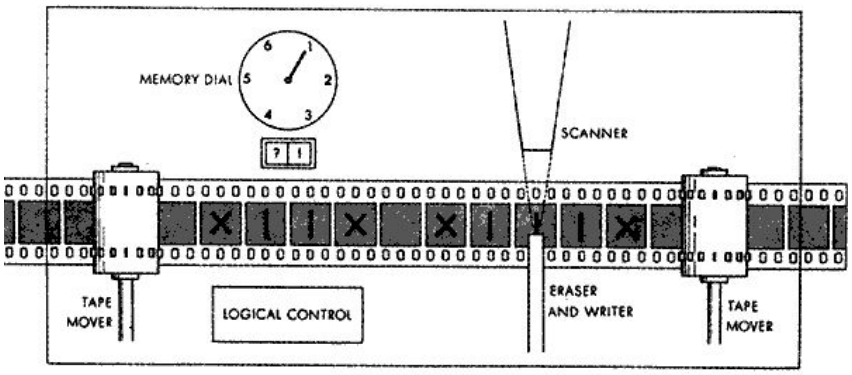
> This system is called "World Computer".

The first mention of the World Computer term was in Vitalik Buterin's project Ethereum. It is implemented using blockchain technology's ability to include executable code into transaction blocks, so every miner's machine automatically executes this code.

Thereby, Ethereum in fact is the World Computer working like a Turing Machine[6], with blockchain used as a state register tape.

This also implies that due to the fact that every program must be run on every machine in the Ethereum network, it is very costly and only a limited range of tasks can be run using this platform.

TURING MACHINE

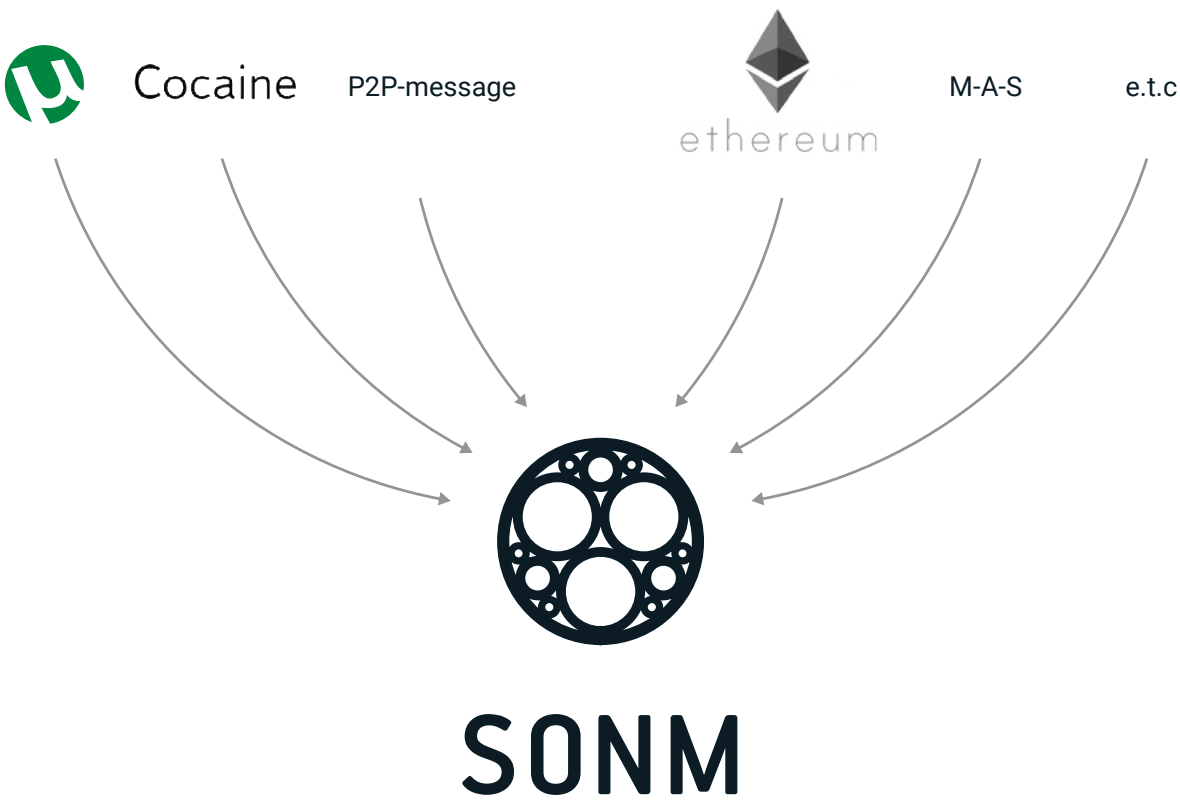


There are other projects which are developing a decentralized world computer (Golem, iEx.Ec and others). It is important to note that all of them are being implemented using the same principles as Ethereum. They also have the same problem: excessive parallelization leading to high costs of operations. This is caused by the absence of any control centers managing task processing in real time that can stop it after receiving the desired result. This in turn leads to running parallel/asynchronous processes.

In fact, these projects can't provide the functionality which any usual personal computer has nowadays.

The SONM team has much experience developing a World Computer functional concept able to process any task, up to the standard of a fully functional computer.

SONM SCHEMATIC

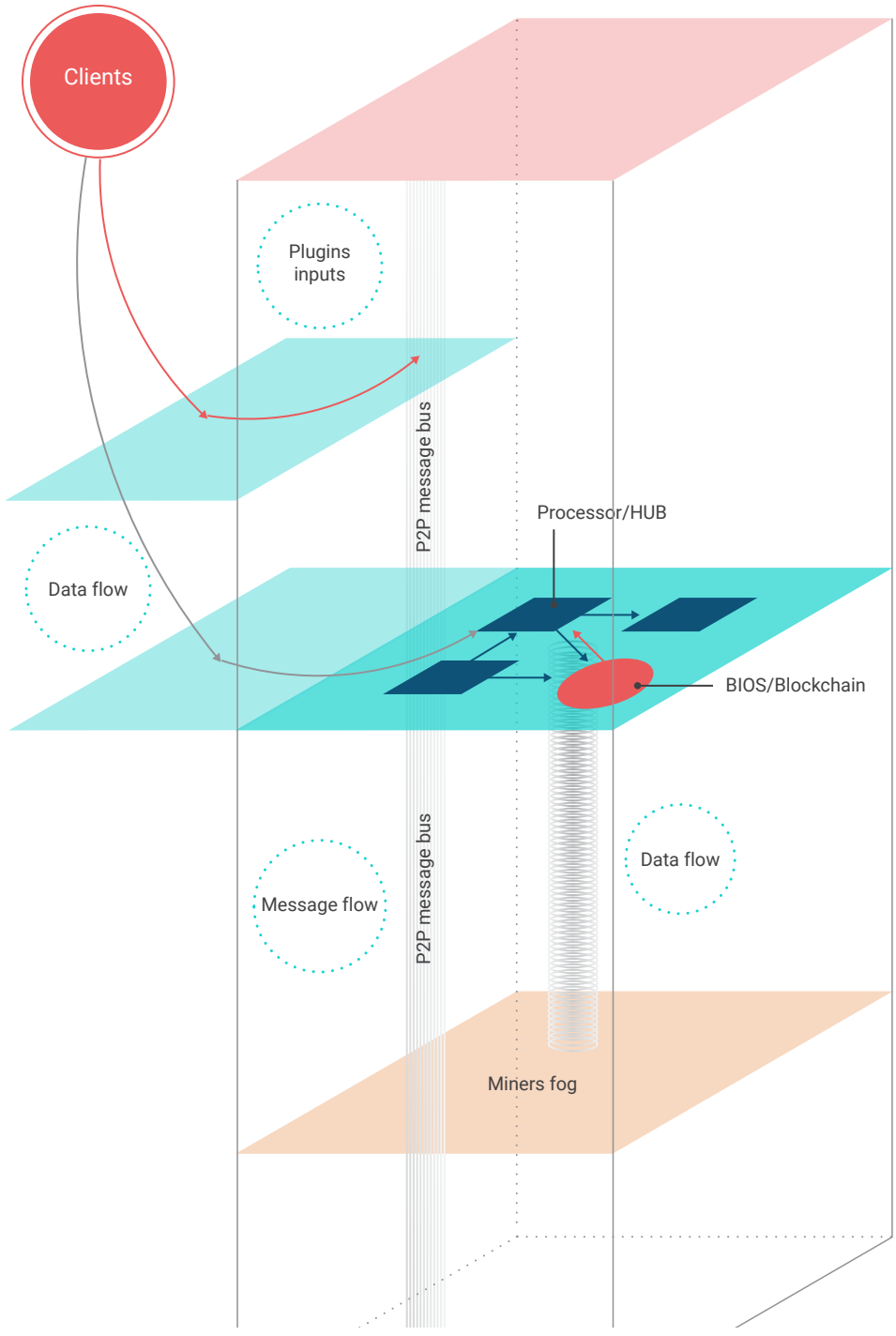


2.3. World Computer General Architecture / Infrastructure

> The architecture of a single computer is the presence of standard components such as CPU, Motherboard, BIOS, Bus, Hard Drive, GPU, RAM memory, etc For our world computer architecture, we decided to follow the modular way all the personal computers are built.

FIGURE: SONM WORLD COMPUTER IMPLEMENTATION SCHEME.

Consider the details of this world computer architecture implementation figure. As you can see, this architecture includes lots of linked elements.



World computer in the scheme has similar elements: CPU, BIOS, bus for data exchange, plugins board (connectable devices), peripheral devices, graphics card, etc.

Hard disk drive analogue will be implemented using decentralized data storage solutions: IPFS (InterPlanetary File System), Storj, Sia, etc.

The first component of the system is the **processor.**

SONM world computer's processor is represented by the set of independent hub nodes distributing tasks, assembling calculations results, keeping statistics and providing uninterrupted operation of the system.

Each hub node on the figure is equivalent to the processor's core (but is not equivalent to the processor). There can be an unlimited number of hubs, and they can be easily included and excluded from the system.

Hubs do not process calculations directly, but rather they represent a very important part of the system, providing management and support (just like a computer's processor regulates and controls the operation of GPU, and is able to process sophisticated high-loaded parallel computations).

Hubs are implemented using Cocaine 'gateway nodes'.

The next element of the system is equivalent to a PC's **GPU**. It is comprised of fog computing miners' processing tasks computations in the SONM system.

The communication bus for transferring data and messages in the network is represented by P2P101 Whisper modified in FUSRODAH communications module protocol of messaging between Sender (Node) and Watcher (worker) machines will be implemented

```
msg := whisper.NewMessage / whisper.Watch
```

https://github.com/sonm-io/Fusrodah

Buyers are equivalent to PC **peripheral devices**, usually used for information input.

The **plugins board** allows the system to constantly expand and gain power by connecting to external compatible networks, for example, any Grid network.

BIOS is an important part of the SONM system, represented by an Ethereum blockchain in our decentralized computer model. As we mentioned earlier, Ethereum systems offer high reliability, but perform only basic operations due to its architecture - this is why Ethereum is the most suitable candidate for the world computer BIOS.

Finally, as we know, PC itself is not worth anything without an **operating system**. Our global computer also requires an OS, and we have it ready.

2.4. World computer's Infrastructure as a service (WC IaaS)

In the previous section we looked at the overall architecture of the system.

The infrastructure part of the system is handled by a messaging framework and a smart contract system (Blockchain government)

2.4.1. Slave Messaging Framework

Currently, the messaging framework is represented by the Slave messaging protocol. (https://github.com/cocaine/cocaine-core/wiki/protocol)

2.4.2. Slave API

Common types

```
Object      ::= <Number> | <String> | <Tuple> | <Map>

Tuple       ::= ([<Object> [, <Object>]...])
```

General format

Every message is a MessagePack-ed tuple of three fields:

```
ChannelID   ::= <Number>

MessageID   ::= <Number>

Message     ::= (<ChannelID>, <MessageID>, <Tuple>)
```

Message ID is a service slot number you're going to call. Every service has its own set of slots which can be inspected by resolving this service via the Locator. Channel ID is a way to multiplex multiple data flows inside a single TCP session. Channel ID is generated by the caller. Tuple is a slot-specific payload.

The usage of Slave will be covered more thoroughly in coming versions.

2.4.3. The smart contract system

The set of protocols described in this section is one of the main contributions of our project. The goal of those protocols is to glue the whole system together and enable trustless but secure interaction of the participants.

We understand that even small flaws in the protocol design may have negative impact on the system integrity, so we decided to invest resources to formally specify and analyze possible participant interactions.

We use applied pi-calculus [7] for high-level but rigorous protocol description. The specification is still work in progress and we consider integrating automated game-theoretic analysis later.

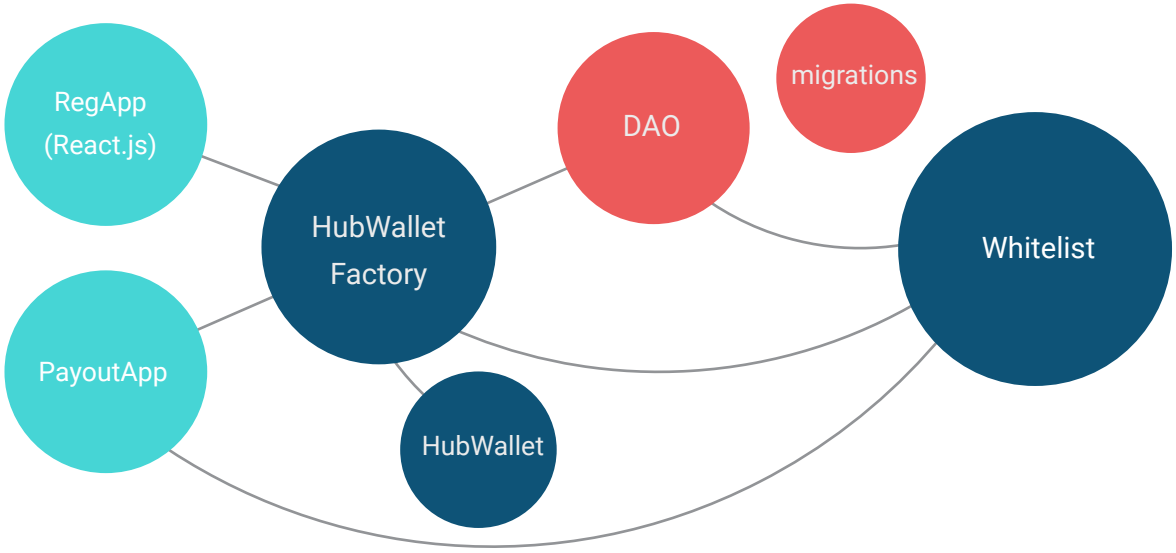
The pi-calculus is a minimal language for describing systems of processes that communicate on named channels, with facilities for dynamic creation of new channels. We use it here without defining it formally, you may refer to [8] for more information.

2.4.3.1. Blockchain government

Blockchain government is an organization (speaking metaphorically) consisting of a court, a DAO, a registry, a factory of enterprises (and an example of said enterprise).

The point of a blockchain government is to provide a simultaneous work process for all enterprises registered in such a system, motivating them to pay "taxes" to the DAO of a higher order, receiving court protection for their enterprise in exchange, as well as protection against unfair partners in the market. SONM uses the following scheme for smart contracts to realize the pattern of a "blockchain government":

https://github.com/sonm-io/Contracts-scheme



Smart contract prototypes can be found here: https://github.com/sonm-io/Forge

Structure of contracts:

1. Migrations(Standard)
2. Sonm Token
3. DAO(Standard)
4. Hub wallet factory
5. Hub wallet
6. Whitelist prototype
7. RegApp (Simple React/Webpack App to work with hub registrations)
8. PayOut App (already implemented for DD@H project) https://github.com/sonm-io/drugdiscovery-token

Abstract

Outline of the smart-contracts system which will be implemented in SONM network is presented. More info about network and contracts interaction can be found in the whitepaper

Simple Data flow

HUB

Before the hub starts paying out tokens to miners and receiving payments from buyers, it must create a hub wallet — a simple contract with a fixed amount of frozen funds. If hub is caught on cheating, DAO can initiate the process of blacklisting this hub and expropriate its frozen funds.

Those expropriated funds will also be frozen at the DAO account for some specified time. This is to protect against malicious decisions of the DAO: tokens can drop in price during freeze, therefore there is no motivation to 'raskulachivat' (expropriate) every hub

```
SNM := !( *({approve, addr_from, addr_to, val}). _
        |  *({transfer, addr, val}) . _
        )

DAO := *

Hub(wallet) := wallet<{deposit}>.
        ( Buyer({x : SNM}) . _
        | _ . Miner<{x : SNM}> . Hub(wallet)
        | DAO(x) . if x == STOP then wallet<{withdraw, val}>. DAO<{freeze, val}>. ()
        )
        | wallet<{setup}>. v(ip) . !HubPool<{set, wallet_addr, ip}>.  _  . Hub(wallet)
```

HUB FACTORY

Hub wallet can be created only by a *Hub wallet factory* (which is actually a simplified replication factory), which creates a new hub wallet contract and registers it in the 'whitelist' contract.

```
HubFactory := v(hubWallet). Whitelist<register hubWallet>. (Hub(hubWallet) | HubFactory)
```

WHITELIST

Whitelist contract is a registry contract containing info about hubs and their statuses. All hub wallets created by hub wallet factory are registered in this contract. It is supposed to be simple registry with a special mapping for 'trusted' hubs. Initially, 'trusted' hubs will be checked by SONM developers manually / official SONM hubs. Later, it's supposed to be also a rating list — everyone could check the hub and rate it (betting some amount of SONM tokens to prevent rating fraud).

```
WhiteList := HubWallet({eth_addr, deposit})
        . v(hub). ( HubFactory<{register, hub}> . _
                | DAO({deregister, hub}) . _
                | HubWallet({freeze}). _
                )
```

REGAPP

As REGAPP we use the React.js application which is simple web application (web-page) with the purpose of user friendly hub registration process.

PAYOUT APP

Payout App is an application to process miners' token payout mechanism operations. For now it is implemented to work with the BOINC statistic mechanism.

2.4.3.2. Example of usage of a 'hub-wallet' contract

Abstract

Before hub starts paying out tokens to miners and receiving payments from buyers – it must create a hub wallet – a simple contract with a defined amount of frozen funds. If hub will be cheating – DAO could initiate process of blacklisting this hub and expropriate frozen funds from it.

Those expropriated funds will also be frozen at the DAO account for some specified time. This is to protect against malicious decisions of the DAO: tokens can drop in price during the freeze. Therefore, there is no motivation to expropriate every hub.

Logic

Contract logic

The contract exists in 4 states - Created, Registered, Idle, Suspected (+Punished)

When the contract is created, the constructor function designates the addresses of the DAO, the factory, the whitelist, the wallet owner and a few other variables, such as the length of the payout period (which is currently set at 30 days). The payout period is a period of time during which the hub can conduct payouts to miners, but cannot take the entire balance for itself.

In the Created state the contract can be registered on the whitelist, freezing a set amount on its balance (1 SONM token). This is designed to circumvent a situation like this – the hub first deposits 0.00000001 SNM, registers the contract, and then deposits the main sum of 100 SNM – the first amount is fixed. Furthermore, the time of registration is recorded when the contract is registered in the whitelist.

After the contract has been registered in the whitelist, it becomes Registered, in which state it has access to the transfer, payday, suspect functions. Let's take a closer look at them in order.

Transfer function

This function enables the contract to conduct payouts to the hub miners. It works as follows: first a lock Fee - is designated, a percentage of the payout which will be locked for the payout period. The default value of it is 30%. Then a limit is set (the total amount of frozen funds + the frozen amount from the registration + the percentage for this particular transaction) and the balance is checked – if the balance is below the limit, this particular transaction is not conducted, if everything is in order – the frozen percentage is added to the total amount of frozen funds and the contract invokes the Approve function (details below) towards the miner. The explanation of why the process is done this way is given in the PayDay portion of the description.

Approve function

This function does not move the tokens to the miner's wallet, but permits the miner to conduct this transaction on his one. This prevents the hub from registering a wallet in the system while conducting the payouts through a separate wallet because the miner is waiting for approval from this particular wallet. Approve is a standard function. (standard ERC20).

PayDay function

This function sets the contract state from Registered to Idle. This function checks the registration time against the current date and thus can be invoked only at the end of the payout period. If this condition is met, it transfers 0.5% of the frozen funds to the DAO wallet, after which it unlocks all the frozen funds and sets the contract's state to idle. In this idle state the contract can move all the funds back to the owner's wallet or register the contract again in the whitelist. During the idle state the hub cannot conduct payouts or be dismantled.

Thus, if the owner can move the funds from the hub to his personal wallet he can do so in two ways – do it in accordance with the rules, wait until the end of the payout period, pay the DAO 0.5% of the frozen funds and move the rest to his wallet; or he can cheat and move all the funds using the transfer function under the guise of paying miners, but in this case 30% of all funds will stay frozen +1 SNM. Such a system motivates the hub to act in compliance with the rules.

The contract also has the Suspected and Punished conditions. In the Registered state – the state when the contract can be registered in the whitelist – the DAO and only DAO can invoke the suspect function, thus setting the contract's stats to suspected – suspected of being malicious. This function blocks **all** funds on the contract's wallet for 120 days.

In the suspected state the following functions can be invoked by the DAO exclusively:

Rehab function

This rehabilitates the hub, removes all fund freezes and set the contract state to idle. Can be invoked at any time.

Ban function

This can only be invoked by the DAO committee after 120 days have passed since the contract's state has been set to suspected. Then all frozen funds of the contracts get sent to the DAO wallet, in which the contract state is set to punished, and the owner of the contract is blocked from conducting further operations using this wallet.

```
HubWallet := Hub({deposit}) .
        . Hub({setup}). _ . WhiteList<{eth_addr, deposit}>
        !( Miner({transfer}). _
        | DAO({payday}). _ . v(val : SNM) . Miner<{fee, val}>
        | DAO({suspect}) . _ . ()
        | DAO({rehab}) . _ . WhiteList<{freeze}>
        )

Miner := !HubPool<select>
        . HubPool({wallet_addr, ip})
        . v(Hub_channel(ip)) . _
```

2.4.4. SONM Miner-Hub interaction solution

Let's consider the process of SONM miners and hubs communicating when they need to establish mutual cooperation (i.e., the first phase, when the miner hasn't decided yet whether to participate in computations and receive tasks from the hub or not).

First, SONM hub administrator sets up an Ethereum smart contract containing SONM tokens used to pay miners for computations.

Then, the ethereum address of this smart contract, address of pool administrator and hub IP are recorded on a special SONM smart contract "Hubs Pool List".

Hubs pool list includes unconfirmed (unverified) hubs and verified hubs (i.e., listed in the hubs whitelist).

The whitelist will be managed by members in the Decentralized Autonomous Organization. In any case, hub information in SONM smart contracts includes the address of the hub owner, the address of the hub wallet and the hub IP. In case of IP or wallet address change, the hub owner can change the hub record.

Therefore, SONM hub records the address of smart contracts containing the funds used to pay miners for computations (so miners can check the existence of these funds) and registers basic information about itself, including the address of the owner and IP.

Then, SONM hub agent starts broadcasting to the network using P2P messenger protocol, sending a broadcast message about itself in the format: «IP, hub owner address, wallet address, hub name».

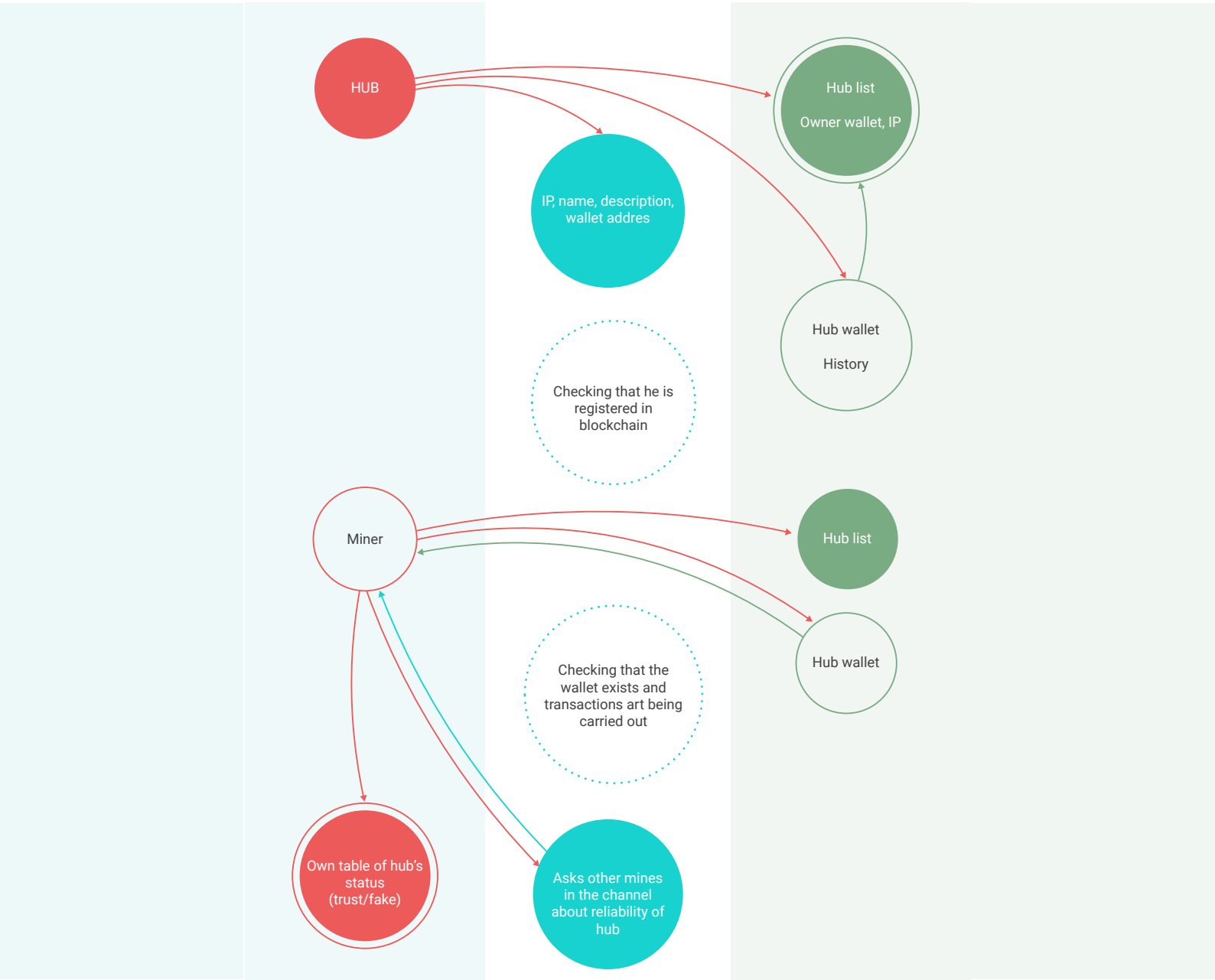
The agent on the miner's side listens to the channel, receives data messages from the hubs, and then makes a request to the Hubs Pool List smart contract to compare the data from the hub messages with data in hubs whitelist. The miner may customize agent settings to accept messages from all servers or only from proven ones listed in the Hubs Pool List.

After that, miner's blockchain agent requests information about the contract-wallet of the hub, amount of funds in the hub's wallet and recent transactions of the wallet.

An intelligent agent checks the received data to compare it with conditions set by the miner. Are there sufficient funds in the hub wallet? Are hub payments to miners regular? What is the average amount of tokens paid to miners by this hub?

Then, P2P messenger agent send a direct message to the hub to request additional meta-data, and records full information about the hub in its hubs list with a "not confirmed" mark.

FLOWCHART OF "MINER-HUB" MESSAGES EXCHANGE:



At the same time, the P2P messenger agent constantly broadcasts question messages to the common miners' data channel for information about the hub, the average amount of reward paid to them, and so on. Other miners' agents broadcast positive answer messages to the channel if hub information in the question message is correlated with their information, or negative answers, if they believe this hub is malicious or not reliable.

If a miner's agent receives a sufficient amount of confirmations from the network, the hub receives "checked" status in the miners' hubs list. If the transaction received by the miner from this hub corresponds to the original agreement, the status of this hub changes to "safe".

After that, depending on the settings of miner's software, a miner can either manually select a hub to connect and perform computing tasks, or a miner's agent can automatically select a hub offering maximum profit and connect to it.

Then, P2P messenger agent send a direct message to the hub to request additional meta-data, and records full information about the hub in its hubs list with a "not confirmed" mark.

At the same time, the P2P messenger agent constantly broadcasts question messages to the common miners' data channel for information about the hub, the average amount of reward paid to them, and so on. Other miners' agents broadcast positive answer messages to the channel if hub information in the question message is correlated with their information, or negative answers, if they believe this hub is malicious or not reliable.

If a miner's agent receives a sufficient amount of confirmations from the network, the hub receives "checked" status in the miners' hubs list. If the transaction received by the miner from this hub corresponds to the original agreement, the status of this hub changes to "safe".

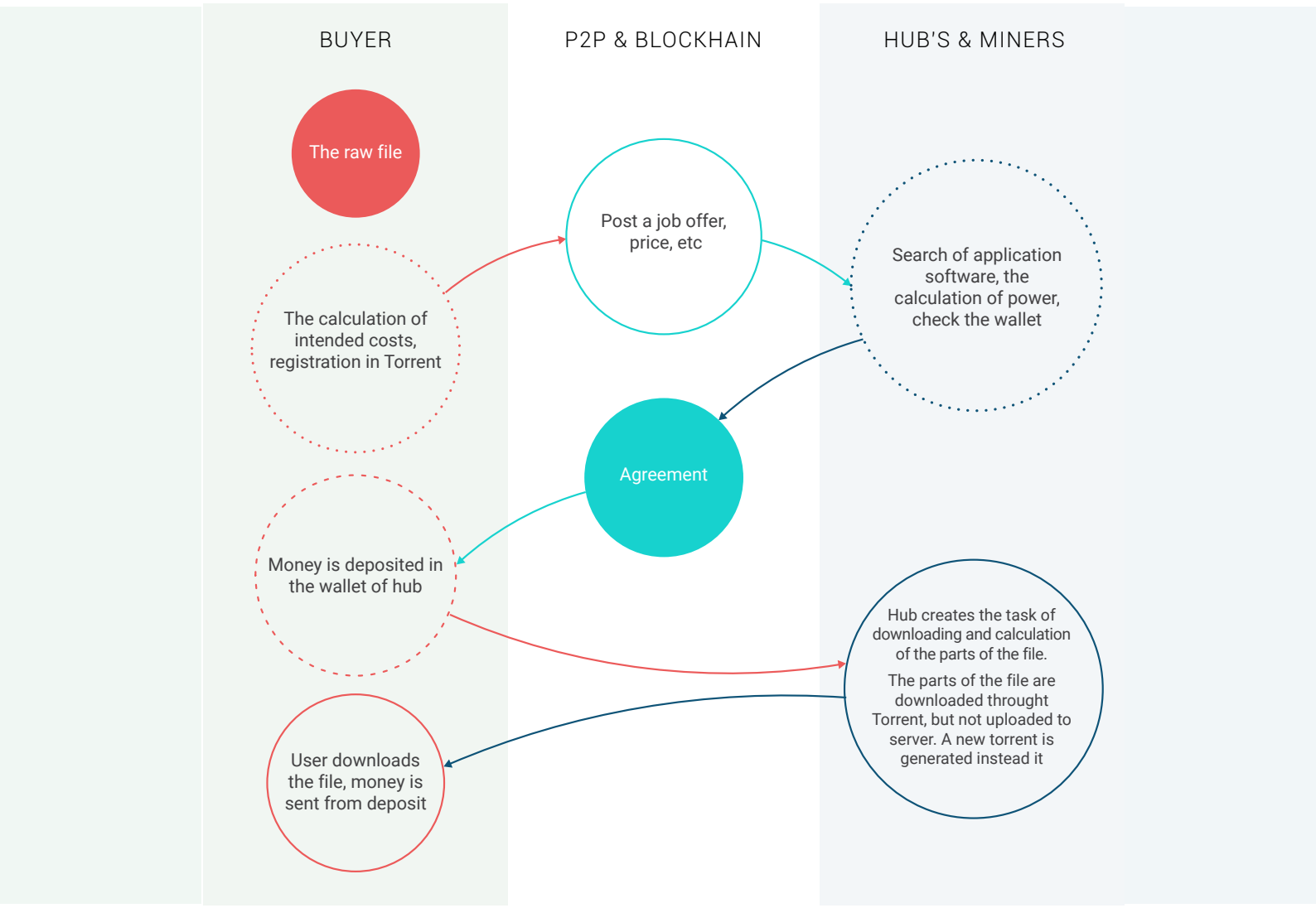
After that, depending on the settings of miner's software, a miner can either manually select a hub to connect and perform computing tasks, or a miner's agent can automatically select a hub offering maximum profit and connect to it.

2.4.5. SONM Client-Hub interaction solution

Clients' (buyer's) interaction with SONM hubs is similar to the miner-hub agent's interaction, with a difference in intellectual agent results' parsing, which for buyers prefers the hubs with the lowest computations price (and vice versa for miners). Buyers will most likely use the "Application Pool" (described in the section 2.8.3.), than "Hub Pool" smart contract.

Buyer creates a task and deposits funds to the hub's smart contract wallet to pay for the job. When the buyer receives the calculations result, he confirms the transfer of money using the smart contract;s function (similar to Multisignature Wallet).

FLOWCHART OF CLIENT-HUB INTERACTION PROCESS:
(Some intermediate messages in the flowchart are omitted)



2.4.6. SONM 'Blockchain-government' Expansion Policy

We previously looked at ways to implement the "blockchain-government" to work with the SONM system using computational hubs as enterprises and miners as "workers", but what if we go beyond the computational model and look that the current smart contract system in a broader sense?

What if we take a random business and try to apply it to the current system? Suppose you are an owner of a restaurant – in which case you can similarly deploy a hub contract on the blockchain and register in the whitelist, while carrying out your regular business transfers – receiving payments from clients and paying your workers, but your bookkeeping will be relatively transparent for anyone, you will be under protection by a DAO (a joint-share group of regular people which will resolve issues via voting),and your business will be registered in the whitelist, similar to the governmental registry, giving your business a "Legitimacy certificate" of sorts and giving you a competitive advantage.

Creating the "blockchain-government" system is not the priority for SONM, but as you may recall, SONM is an assembly. We suppose that those interested in the system described above will register on the SONM whitelist, thus executing the expansion plan for the "blockchain-government" into other markets and implementations.

2.4.7. SONM Client-Hub content delivery method

Content delivery method is the only significant difference between client-hub and miner-hub interactions.

As you might expect, there is no difference between rendering a 6-hour video using the local computer and uploading this video to the server while waiting for video rendering on the remote server, because most of the time will be spent on uploading.

We developed a solution for this issue:

When a client wants to upload a large file of raw data to the server, SONM automatically creates a torrent and sends a message to the selected hub. This hub receives the message and creates a task sequence for torrent downloading, computation work with downloaded file(s) and creation of a new torrent for calculation results file.

After processing the calculations and creating a torrent for the resulting data, the hub sends a message to the buyer, who only has to download the received file from the miners.

We expect this to be the most rapid solution of all those that exist at the moment.

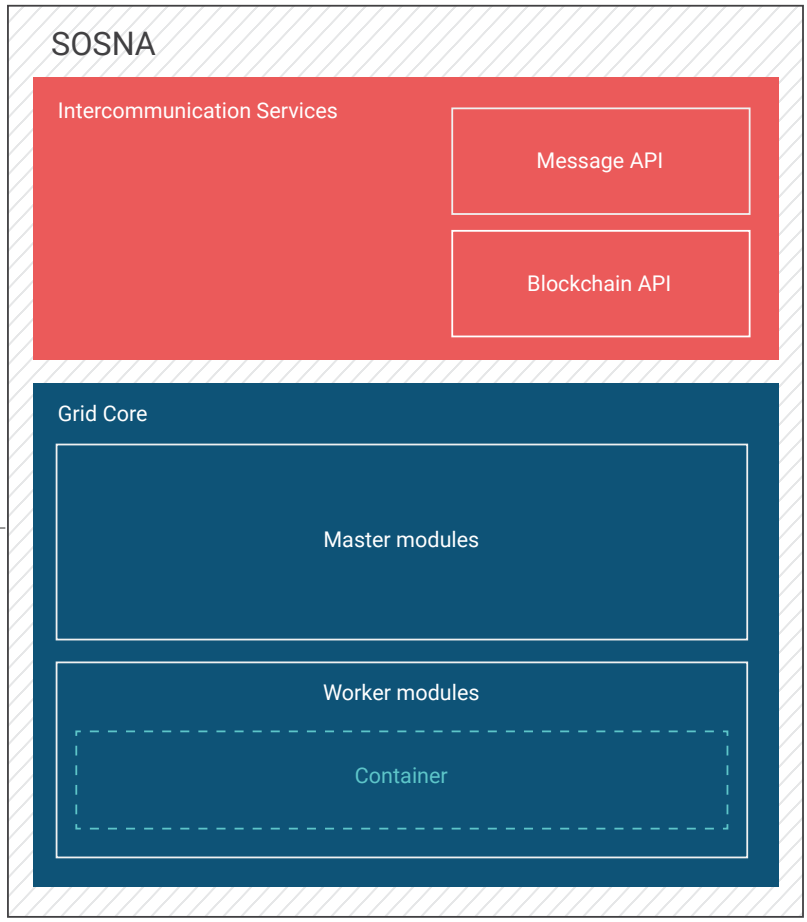
2.5. SOSNA in a nutshell

As a platform for SONM we propose using SOSNA – Superglobal Operation System by/for Network Architecture. (look at the scheme on the next page)

2.5.1. What is SOSNA

SOSNA is a global operating system built on the nesting doll principle. It is important to understand the structure of SOSNA in order to internalize this concept. Let's go from the end-user application to outer-layer infrastructure. SOSNA itself is a top layer envelope that works with the Grid-core (BOINC, Yandex.Cocaine/ Other grid-compatible PaaS) and the infrastructure of SONM smart contracts

SUPERGLOBAL OPERATION SYSTEM BY/FOR NETWORK ARCHITECTURE



2.5.2. Applications and containerization

When you are developing an application, you must make sure it's will function properly that for the end-user. But if the end-user's computer does not possess the same amount of libraries as the one you

used during development, or they may not be up-to-date with vulnerabilities left, it may cause unexpected results. Is there any way to force the program to run exactly as it was intended, and at the same time make it run safely for the end-user? For this, there are containers.

Containers allow us to run * any * software inside a secure, isolated environment. By itself, such a container is a miniature virtual machine, packed with all the dependency libraries of your system - so the compatibility problem and dependencies are relatively resolved. In addition, such a system is isolated in relation to the host system, so no one can cause harm to the miner's computer.

(link to 2.8 safety and security)

2.5.3. Slaves & their services

Let's move up one level. Miner's Host in this architecture is a simple node, a worker. (In cloud architecture such a system is called Slave or Minion). All applications performed inside containers are called services. We will talk more thoroughly about what containers are in the SaaS chapter. The miner's host itself can be definitively represented as an assembly of services and a service location system.

Service

Service is an actor, an RPC-enabled piece of code, which accepts a certain set of messages. Technically speaking, each service dispatches a service protocol — that is, a list of methods and their respective SlotIDs you can call by sending messages to the service just after a connection has been established. This protocol description can be dynamically obtained (along with other stuff) by resolving a service name via the locator.

> *The important part here is that, in line with the actor model, the client is an actor too. So, after you have sent a message to a service to do something for you, it responds by sending messages as well. But unlike server-side services with service-specific protocols, every client dispatches the streaming service protocol, mostly for backward compatibility and ease of use.*

Each connection between a client and a service is multiplexed using ChannelIDs, and both ends of a given channel dispatch some specific, possibly different, protocols. For example, the usual session between a client and a service goes as follows:

- A client connects to some service and picks any channel at random (for example, channel #1), because all of them are not used in the beginning. Initially the service side of a channel dispatches the service-specific protocol, and the client side dispatches the streaming protocol.
- The client sends a message tagged with the chosen ChannelID in order to call one of the service's methods. That indicates the start of a session.
- he service switches its side of the channel to the null protocol, so that the client couldn't call some other method in the same channel while the service processes you request.
- The client starts to receive the streaming protocol Chunk messages with the service response.
- In the end, the service sends a Choke message to indicate that the session has been completed and switches its side of the channel back to the service-specific protocol.
- If that was the only request, the client disconnects

Note that some services provide streamable methods: in that case the service will switch to the streaming protocol instead of the null protocol, so that you can stream some data to the service.

Locator

When a node starts, it reads its configuration file, which has a list of services to run. This list only specifies service names and types, but not network-related properties, because the I/O layer and the RPC layer are completely separate. Moreover, the services themselves have no code to communicate over the network, only the message dispatching code.

In order to enable those services to receive and send messages over the network, the node starts a special service called the locator. Every other service is attached to the locator, which in turn wraps them in an event loop, binds them to some network endpoints and announces them in the cluster. The locator itself always runs on a public port.

So, a client should perform the following steps to connect to the requested service:

- Connect to service locator on a public port.
- Send a Resolve message with the name of the required service using any channel.
- Receive a Chunk message with the information about the service endpoint, its protocol version and its dispatch maps (which is a mapping of message numbers to method names).
- Receive a Choke message indicating that the request has been completed.
- Connect to the specified endpoint and work with the requested service.

Services can stack protocols. For example, the Elliptics service implements both the generic storage protocol and its own specific protocol, which means that a client requesting storage service can be routed to the Elliptics service instance. That is fine, because stacking allows the client to work with the Elliptics instance without even knowing the service-specific protocol details — protocol messages have the same SlotIDs no matter what service implements the given protocol and whether it uses protocol stacking or not.

2.5.4. Masters and Gateways

Let's advance one layer higher. Here you can see that in addition to the miner machine itself, there is a master machine, i.e. Hub, the approximate function of which we considered in the paragraph about IaaS.

Master

Master manages the execution of services on the machines of miners, maintains statistics, balances the load, carries out the validation of results, leads the task planner, etc. - i.e. Behaves like a conventional cryptocurrency pool. Master is also called a Gateway node.

Gateway

Optionally, the locator can be configured to aggregate other locators' multicast announcements (or use a provided list of remote nodes) and act as a cluster entry point for clients. In other words, the aggregating locator job is to configure a gateway by connecting with all the remote nodes and monitoring their health and service updates.

Gateways are pluggable locator modules which provide remote location functionality. For example, a simple builtin Adhoc Gateway randomly picks a remote node for each client, and IPVS Gateway operates on a kernel IPVS load balancer to set up a local virtual service for each available service in the cluster.

Clients can use these aggregating locators to access every service in the cluster regardless of their physical location in a load-balanced fashion.

2.5.5. Grid - Core

Two machines - Master & Worker form a basic implementation of the Grid standard - a loosely coupled computing network. A key feature of the Grid standard is the prerequisite for decentralization and geographic remoteness of Masters from Workers. As an example, we consider the product https://github.com/cocaine/cocaine-core as an example of Grid-Core.

2.5.6. Intercommunication Services

SOSNA intercommunication services are a common p2p message bus, with which miners, hubs and clients communicate, as well as the Blockchain API service, which allows SOSNA to communicate with Blockchain.

2.6. World Computer SaaS and its API

Example of the simple application that can be run on SOSNA

```
#!/usr/bin/env python

from cocaine.services import Service
from cocaine.worker import Worker

storage = Service("storage")

def process(value):
    return len(value)

def handle(request, response):
    key = yield request.read()
    value = yield storage.read("collection", key)


    response.write(process(value))
    response.close()

Worker().run({
    'calculate_length': handle
})
```

2.7. Results verification

The problem of validating computations executed by a third party is a thoroughly researched topic [9] [10], but it still lacks production-ready solutions, since most of them are very expensive in practice (at least in an HPC setting).

More practical solutions are based on repeating computations – verification by replication. This approach requires designated nodes (hubs) to distribute work units, aggregate results and verify them. Docker uses this approach and has a highly tested implementation.

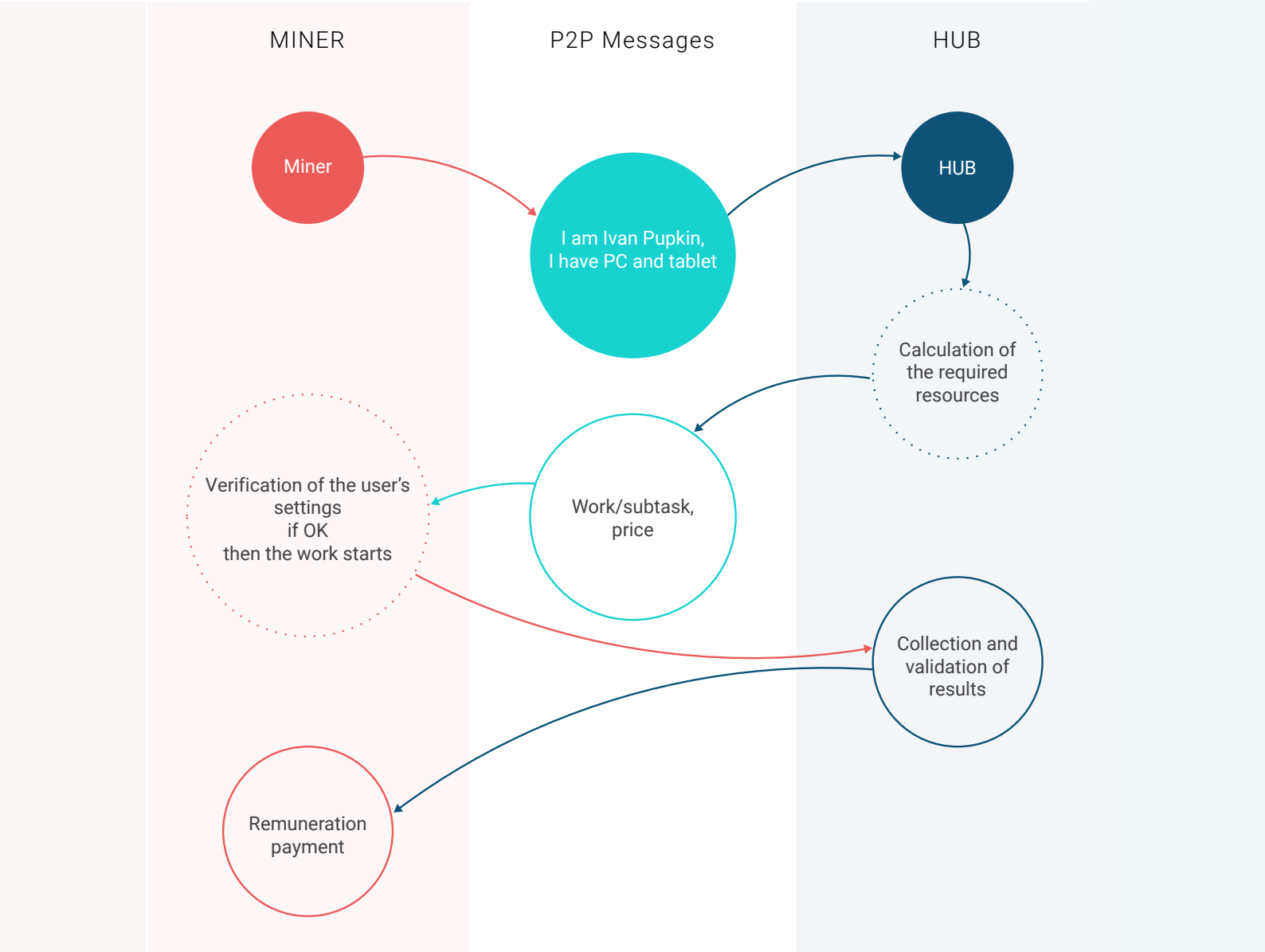
For certain kinds of computational problems it may be practical to offload the task of aggregation and verification to a smart contract. The process goes as follows: miner computes some work unit and posts

merkle-tree root hash to the smart contract. Some other miner computes the same work unit and notices that results differ. In this case, it is possible to calculate a compact proof of cheat. The proof can be checked by the smart contract, and the cheater punished.

Economic motivation is used to promote this double-checking behavior: miners deposit some fixed amount of tokens, and this deposit will be returned after some timeout if no proof of cheat was posted. On the other hand, it is possible to earn tokens by checking computations and revealing cheaters.

Verification by smart contracts is actively researched [11][12] and has some benefits:

- *does not require trusted third party to aggregate and verify results*
- *does not impose any overhead in case of honest miners*
- *has limited and bearable overhead in case of dishonest miners*



Notes:

We will have a fully functional system that will be used for any general-purpose computations, starting from the v.2.0. It is most likely that by this stage the SONM platform will have full-scale computational projects deployed with high turnover volume. (link to chapter 3. Roadmap)

Moreover, by the v.2.0 we expect SONM to attract lots of open-source community members, which means:

- The community will be independently creating lots of decentralized grid-compatible apps.
- Lots of brand new markets and teams are potentially going to appear, as well as numerous community-crafted tools for interaction with the SONM platform, most likely better than the original apps, developed by the SONM team. For example, the official geth Ethereum client made by Ethereum Foundation comparing to Parity by EthCore, or Windows Media Player compared to WinAmp or Internet Explorer comparing to Mozilla Firefox. We understand and welcome it

That means that starting from this point we will need to reduce our efforts for tools development and give way to the free market and community.

We will focus on creating new formations for interaction with this market:

- a dedicated team developing decentralized computational power exchange
- teams providing server hosting services based on the SONM platform
- software for niche markets
- various integration projects
- external formations for tools development (like Metamask.io by ConsenSys)

I.e., by this point we will have a distinct division of SONM development areas. For example:

- original SONM core developers are creating basic protocols of the system
- another team is creating apps within smart-solutions
- SONM ExChange team is building UI-friendly tools for the interaction with buyers and managing the decentralized exchange

2.8. Safety and Security

> In this chapter, we look at security aspects for miners and buyers

2.8.1. Safety for miners

Safety from hostile workloads. Docker isolation.

One of the docker's software packages is a daemon - which consists of a container server,launched via the "docker -d" command), client tools which permit the user to control the modus and containers directly via the command line interface and an API which permits the user to control the containers via a REST-style program.

The daemon provides a complete isolation for the containers launched on the node at the file system level (each container has his own root), at the process level (the processes have access permission only for the container's own file system and the resources are split up using libcontainer), at the network level (each container has access exclusively to the range of network names tied directly to it and the corresponding network interfaces).

Safety for "reputation" and pirates.

How miner can secure himself from payloads, which don't harm his computer, but have the reputational risk possibility? There can be website from darknet, bot-machine, credit card hacker or porn on his computer, so how our system provides security against such sort of thing?

We use the special 'Whitelist' contract for that, which is part of the Anti-Fraud system, where each hub and miner is registered and each of them, who is registered in whitelist, can be blocked and kicked out from the system by DAO voting. The pirate contracts of hubs and miners' wallets, which are not registered in the whitelist, will not be visible from the general settings, so they won't be affected by DAO voting. Therefore, hubs will have to offer to host only such kind of services that meet the community reputational requirements, which are formed by the community itself. It is also obliviously that one can configure the miner client program to accept the proposals from the pirate hubs, but then there are no any guarantees of security and fair payment.

2.8.2. Dishonest nodes eliminate

The contract also has the Suspected and Punished conditions. In the Registered state – the state when the contract can be registered in the whitelist – the DAO and only DAO can invoke the suspect function, thus setting the contract's stats to suspected – suspected of being malicious. This function blocks all funds on the contract's wallet for 120 days. In the suspected state the following functions can be invoked by the DAO exclusively:

This mean that the only way to hub to get all his money including lockedFunds - is to invoke this function and pay DAO 0.5% of lockedFunds. In any other case 30% from all hub's operation's will be locked on the contract balance.

Suspected - Gulag function

This can only be invoked by the DAO committee after 120 days have passed since the contract's state has been set to suspected, then all frozen funds of the contracts get sent to the DAO wallet, the contract state is definitively set to punished , and the owner of the contract is blocked from conduction further operations using this wallet.

2.8.3. Safety for buyers

How is ensured to honest buyers their task is being run for given time?

The first method is by query metrics. Hub should understand how much resourses are consumed by one instance of task, depending on the number (for example) of the input connections. Therefore, Hub can estimate the one connection value and then estimate work by its number. Of course, it gives just an approximate result and works only with stateless tasks, where users don't transfer the complete data volume into application (or we know in advance the approximate data volume). I.e. relatively talking, it will work for the casual websites and applications, but, for example, for such kind of service as photo and video editing

```
client - > buyer(hub) -> miner
           quieries ------> miner
           hub <----------metrics
```

The second method – by resources. We should make a request inside the platform or container to specify how much resources were spent. However, there is a danger that miner can open everything on his computer and fake the data. In contrast to the number of requests that we can measure on our Hub machine, this method is no longer reliable (although, more accurate).

The third method is reputational. We assume that networks will function both like “fog” public parts, i.e. like clusters from multiple independent solo miners, and like private pools, i.e. private clouds (or probably some informal unions of solo miners like, for example, cartels).

In this case client wants to choose, does he want to run the task in the multiple independent and difficult to verify nodes, or in the private nodes (private hubs).

How is ensured to honest buyers their task is being run properly

We take container and put out the hash from it. If hash coincides with mastercopy on the hub side, we decide that everything is good and it is our container. Hash is put out by the Docker engine and subscribed by the platform private key, which is unique for each installation. Then, among other things, we also force the Docker engine and platform key to update, so potential hacker will have to learn how to hack the whole system very-very fast and, in fact, with such a security system there is zero possibility of any successful hack.

In addition – PoS and reputation – every miner, who wants to earn more, must make a deposit on his wallet as a proof of “decency”, guarantee.

Also, it is possible to contact directly the launched container by ssh and check the ongoing process.

2.9. AI implementation

Artificial intelligence is implemented for three main tasks based on various properties and regularities of algorithms and various formal models of their representation. First of all, it is worthwhile to focus on the task that solves the “trust problem” in relation to specific hubs. Methods implementing artificial intelligence, namely, methods of computational AI, such as neural networks and methods of evolutionary computation are capable of contributing to the solution of this problem.

Based on the logging of the messaging channel and receiving certain information about the hubs and their characteristics, an image of “natural selection” is created, at which the least optimal sets of characteristics are eliminated according to the specified decision criterion. The program analyzes the rating system of the hubs and their activity and creates a request to the blockchain to confirm. Only after an answer has been received it gives consent (with the successful state of the channel and a high level of trust) to connect and make a deal with this particular hub.

The neural network is based on the data filtering algorithm (takes into account OLS), uses a binary decision tree and clustering algorithms (used to extract information, compress data and examine data properties).

As a result, with the help of neural networks SONM analyzes and predicts the risks of transactions with each particular hub and solves the trust problem.

As Goolge (ex DeepMind) team show in 2016, using of AI can significantly improve loading of data center and got up to 40-60% reducing energy consumption in their datacenters . Going this way, to reduce costs (thus get up profit) for miners, SONM, using wide number of advanced metrics from master (“Hubs”) and slave (“Worker”) machines, aggregate this big data and train Hubs NN-based AIs to optimizing dispatching of data to Workers. We’ll train AIs to solve Combinatorial optimization problems [13], for example The Knapsack[14] problem and the Travelling salesman problem [15]. These problems are NP-incomplete, so we have a basic implementation of a weak-class AI.

The Knapsack problem is solved in the context of ‘miners briefcase’ - how to divide resources between different projects/hubs, with maximal profit and risk diversification. Put simply, it would be like “What coins do I need to mine if btc goes down and in what proportion for each of it?”

The Travelling salesman problem is solved in the context of resource distribution and backs to GRID-network standards (this feature is not fully implemented yet).

2.10. SONM GitHub repositories

github.com/sonm-io

2.11. UI and API

2.11.1. Example of how the SONM marketplace works

Market mechanisms.

One must remember that the end-user rarely interacts with the market directly, the market is mainly used by owners of hubs, miners or developers.

An overview of the marketplace from the viewpoint of the Buyer (Developer).

CLIENT-HUB POINT OF VIEW:

1. Hosting applications

This is represented by a standard market mechanism functioning as a cloud market: an aggregator, where users can choose the cluster to host their applications. The cluster is chosen depending on the preferences of the client : region, pricing, power, etc. The system basically functions like the market.

This diagram represents a simple market mechanism, where a developer picks a cluster where the app will be hosted (a few clusters can be chosen) .

2. Selling utility services

A developer, which has created a utility service, can sell it or delegate it to hubs, collecting passive income via it. An example of such a service can be SUBD, a messenger service which is registered in the Application Pool and is offered to hubs. Hubs can be interested in using said application to attract more clients and gain advantage over their market rivals, which leads to market growth. The developer can view the service statistics in his personal account.

An overview of the marketplace from the viewpoint of the Worker (miner)

HUB-MINER POINT OF VIEW:

Miner, according to his preferences, automatically, according to set criteria, connects to the hub which mays the most and has the most stable bandwidth. As such there is a market here as well – on one side there is the computational power, on the other – money, but this market is almost entirely automated and unnoticeable to the miner (the system was designed to let the miner simply press a button and not monitor the rest of the process).

To summarize:

For the client everything is easy – his application will be run on the closest miner. This means that for the end-user the difference will be virtually unnoticeable. The client/owner will be paying less, and the application will run faster.

On the technical level the application will first request the designated hub or the required amount of computational power (in order to contact the locator service) for service, and if such a service is available, the application will request it to be performed on the closest machine available (which will accordingly be the cheapest option).

2.11.2. Interface prototype

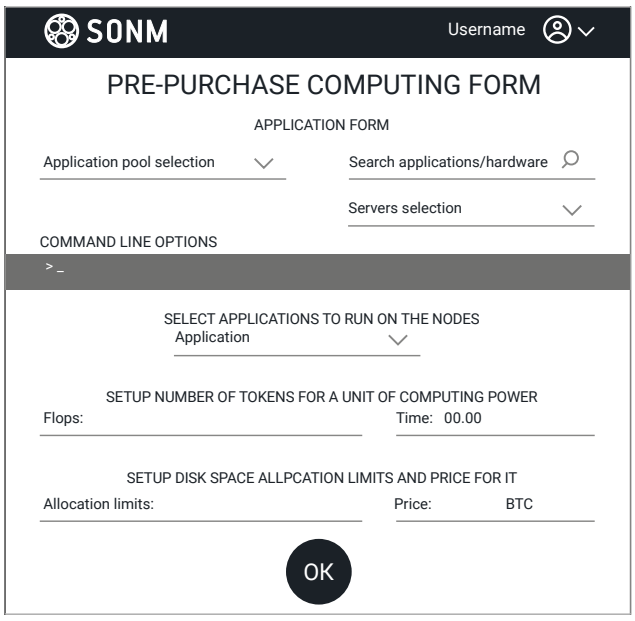
SONM buyers interface prototype

Pre-purchase computing power

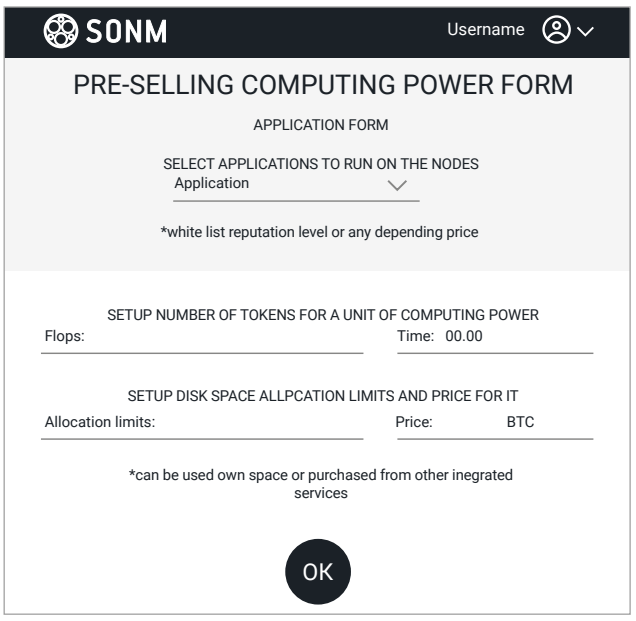
- Hardware selection
- Application from Application Pool selection
- Search for applications/hardware
- Servers selection
- Command line (Terminal interface) for running own application, setup running application on the server, before it is sent to miners nodes

SONM miners interface prototype

- Select applications to run on the nodes (whitelist, reputation level , or any depending on price)
- Setup number of tokens for unit of computing power (FLOPS, time)
- Setup disk space allocation limits and price for it (can be used own space, or purchased from other integrated services, like Oraclize, Factom, Storj, Sia, Filecoin etc.).



SONM BUYERS INTERFACE



SONM MINERS INTERFACE

2.11.3. API for software developers

In the first stages developers' API will be implemented using widespread and well-tested Yandex, Cocaine and Ethereum API.

In the further stages of platform development, after the system core upgrade to v.2.0, we'll create proprietary SONM API.Top-level api, which defines the logical grouping of containers, which allows you to define container pools, distribute the load, and also specify their placement.

3. DEVELOPMENT ROADMAP

3.1. Modules' implementation roadmap:

Ver.	Messaging	SOSNA core	Platform	Smart-contracts
0.1	Slave Protocol	-	Yandex.Cocaine	PresaleToken, Presale, "Factory"
0.2	Ethereum P2P101 Whisper modified in FUSRODAH - Messaging protocol implementation	-	PayoutProto	ICO, Token
0.3	Messages optimization (channels,anti-flood etc)	Waiting for contracts' deployment. Interaction protos	Debug + Cutting off Yandex pitfalls.	DAO
0.4	Sonm hub DNS reconstruction, additional messaging types and channels' specification, debug	Business logic implementation (including price API)	Payout Dapp	"Factory" debug
0.5.	Debug and feedback	Interaction with p2p message bus & ethereum blockchain API	DCFS (etcd, Swarm, IPFS) integration	Whitelist, Hub wallet, Hub Factory
1.0	Global channels and Global DNS improvements.	Graphic UI	Locator service improvement.	BugFix + Escrow
1.1	Debug & feedback	Debug & feedback	BugFix	Debug & feedback
1.n			Debug & feedback	
2.0				
			CoreOS (https://coreos.com/)	

v.0.1 - ANGE (Current Version)

Yandex. Cocaine as a platform, Docker as an isolation.

Supported languages:

- C++
- Go
- Java
- Node.js
- Python
- Ruby
- [In development] Racket

We have the following services:

- Logging
- Node-local file storage
- MongoDB storage
- Elliptics storage
- Node-local in-memory cache
- Distributed in-memory cache
- URL Fetch
- Jabber
- *[In development]* Notifications
- *[In development]* Distributed time service

Prototypes of the smart-contracts system (“Forge”), Slave protocol for communication between nodes.

Anyone can create his own hub and try to collect powers from miners, or create his own cluster (from many owned machines). Anyone could run any usual docker container on it or create your own application in Cocaine framework (see sections above or github).

v.0.2 - PRINCIP (june 2017)

Main Token Contract and ICO application. Payout prototype (already implemented for BOINC-platform “DrugDiscovery@home”)

Optimized To>man - Yandex.Cocaine - based platform for fog computing will be implemented.

Ethereum P2P101 Whisper modified in FUSRODAH - based protocol of messaging between Sender (Node) and Watcher (worker) machines will be implemented.

As a demonstration of universality of platform, QUAKE game server container will be implemented .)

v.0.3 - ARCH (september 2017)

On this version we will focus on the most crucial parts of the system. For now the slave protocol is literally ‘protocol’ - it has no own libraries or API, it’s just an agreement inside the module system.

On the platform level it is will be PayOut dapp - a simple dapp which allow hub administrator payout tokens to miners, depending on their work - it is already done for BOINC-like platforms such as“DrugDiscovery@home”. We need to simply adapt it to our newly deployed token contract and architecture of the Cocaine gateway node.

On the smart contracts level it will work with our DAO contract.

+Browser for sites

+Messages optimization (channels,anti-flood etc)

+DAO & Dividents payout

+ UI and UX improvments for all sides.

+General productivity improvments

Browser for sites means we will have our own browser for our services and sites (inside sonm network). Probably it would be web3 compatible (metamask in complect) - It’s also will be good test for locator services - network should spawn apps and sites nearby with browser user (cookie definition and e.t.c.).

v.0.4 - POWER (march 2018)

In this version we will be adding new messaging types for new messaging systems, tuning the communication between miners and hubs. We will probably rewrite the internal DNS peer discovery service as well (it allows the searching of peers during listening of the general channel in the messaging system).

Concerning the core platform we will work towards business logic (market and AI) implementation, and tuning in messages and blockchain API.

On the platform level we will be implementing integration with DCFS like IPFS,Swarm, Storj. Application registry. On the smart-contracts level we will be finishing work on “Factory”.

v.0.5 - VIRT (june 2018)

Application registry means some ‘trusted’ registry of approved applications. If miner or hub will want use applications not from this registry (e.g. using their own private registry or other sources like github or private dev repositories) - they should add new source manually etc.

Court app - is application which filter DAO proposal and parsing messages channel to structurise flow of ‘blaming’ proposal and let community better tool to look at it. And vote for it.

On this level all the newest contracts from “Factory” - Whitelist, HubFactory,HubWallet will be deployed. It will be the start of forming real new homeostasis of the system. After that we think that a few “debug” releases will be necessary with different community proposals.

+General locator and end-node client

This mean that locator will be used not only for browser, but for any type of application. We think it will take all queries from localhost to the network services and look for this services in sonm network to determine nearest good point.

+BTSync

+Court app

v.1.0 - DOMIN (august 2018)

The very first commercial version of this platform for public usage.

Global DNS and service locator improvements allow us to create a new internet browser, which would allow everyone to find and run services like https://servicename.

Graphic UI improvements for each part of the system permit us to improve the user experience and start to widely expand among 'non-bitcoiners'.

We also believe that other companies will use our smart-contract's organization (Forge), which would allow them to use one contract-register and fair system protection from malicious users and fraud.

v.1.1 - THRON (november 2018)

UX improvement, community proposals, feedback, debug, etc.

v1.n - CHERUB (2019)

Development of the new SOSNA version is started, which will be based on CoreOS (a system you could literally run everywhere - microwaves and washing machines). *Seriously, read about CoreOS - it is awesome!*

v.2.0 - SERA (2020)

Release of SOSNA 2.0. Imagine if your smart-watches from Apple could earn you money? That's what we are talking about - When "Time is money!" is not just words.

Notes:

We will have a fully functional system, able to be used for any general-purpose computations, starting from the v.1.0. Most likely, by this stage SONM platform will have full-scale computational projects deployed with high turnover volume.

Moreover, by the v.1.0 we expect SONM to attract lots of the open-source community members, which means:

- The community will be independently creating lots of decentralized grid-compatible apps.
- Lots of brand new markets and teams are potentially going to appear, as well as lots of community-crafted tools for interaction with the SONM platform, most likely better than the original apps, developed by the SONM team. For example, the official geth Ethereum client made by Ethereum Foundation comparing to Parity by EthCore, or Windows Media Player compared to WinAmp or Internet Explorer comparing to Mozilla Firefox. We understand and welcome it.

That means that starting from this point we'll need to reduce our efforts for tools development and give way to the free market and community..

We'll focus on creating new formations for interaction with this market:

- a dedicated team developing decentralized computational power exchange
- teams providing server hosting services based on the SONM platform
- software for niche markets
- various integration projects
- external formations for tools development (like Metamask.io by ConsenSys)

I.e., by this point we will have a distinct division of SONM development areas. For example:

- original SONM core developers are creating basic protocols of the system
- another team is creating apps within smart-solutions
- SONM ExChange team is building UI-friendly tools for the interaction with buyers and managing the decentralized exchange

3.2. Dissemination of the development process information

- The project team is responsible for making the results open to the public and for using all available resources to disseminate information about the project.
- We will publish a report about current development results and issues at least once a week.
- Report will contain current project needs and issues.
- All major breakthroughs will be communicated with interested mass media and spread in major community forums like BitcoinTalk and CryptoCoin Talk.

4. SONM IN COMPARISON TO OTHER GRID COMPUTING PROJECTS

	SONM	Golem	iEXEC
Platform	Cocaine	Golem itself	Xtremweb-hep
Anti-fraud	✓	✗	✗
Game servers support	✓	✗	✗
Site hosting	✓	✗	✗
Services	✓	✗	✗
Hybrid P2P	✓	✗	✓
SaaS	Already yes	✗	✗
Messages API	✓	Planned in stone golem	✗
Fog-computation	✓	✓	✗
Non-determenistic tasks	✓	✗	✗
Container protection	✓	✗	✗
Container validation	✓	✗	✗
Load balancer	round-robin	✗	✗
Service locator	✓	✗	✗
Data stream & pipeline	✓	✗	✗
Distributed in-memory cache service	✗	✗	✗
Nodejs application-service support	✓	✗	✗

1. Platform

Grid is a formal technology/a standard of geographically dispersed computational networks.

Let's take a look at how these three projects design their platforms.

Golem are creating their own platform for computations, in our opinion this is similar to reinventing the wheel and has no obvious merit. There's no clear understanding of what basis for the technology are they using (GRID or Cloud), no logical justification for using Python as their main programming language (Python is web-oriented, creating a platform requires more appropriate language, like C++ or Java).

iEX, on the other hand, is based on Xtremweb-hep. The Xtremweb-hep platform is based on the GRID technology and written using Java.

Finally, SONM uses and is going to improve the Yandex.Cocaine platform. Yandex.Cocaine is based on a Distributed Cloud structure (unlike the regular Cloud structure,there is no hard master-minion assignment, like the one used in Kubernetes by Google). It is written using C++, has implementations for Go, Java and Haskell, among others. Our programmer Max Taldykin was part of the team designing the Haskell implementation.

What can be considered the best solution for a world computer platform?

A hybrid technology must be created and used, Grid+Cloud.

Golem cannot provide a competing solution at this moment, their own projects are far behind the most innovative ideas. Their popularity is largely driven by the fact that they were the first ones to showcase the distributed world computer structure, and now the only way forward left for them is to start working on restructuring their work algorithms and integrating new technologies.

iEX has a few very good solutions presented on the technical level using the Grid technology, but most Grid solutions are based on scientific calculation needs and have poor compatibility with traditional clouds. They also lack locator services and many more. Another disadvantage of their technologies would be the need to work in the direction of the Cloud technology (developing system messages, metrics, accounization, billing, services).

SONM is based on the Distributed Cloud technology and has all the necessary resources to create the proper platform. We are planning to adopt a few technological distinctions of the Grid system – integrating an additional validation system (it is already being prepared and in soon we will present a unique system of undetermined computation verification). There are no similar projects developed and copying it will be very difficult for competitors. Further development of existing containers with the intention of unifying processes computed by the miners is also planned.

2. Anti-fraud protection

One of the most important aspects involved in the effective implementation of distributed computing is anti-fraud protection.

SONM has a working prototype of a unique anti-fraud protection system based on smart contracts. This will provide miners and customers solid protection from malicious nodes, clusters and pools within the system. This system is also a key to us upscaling, since this system can be used as a separate product.

You can find more information about the smart contract system here:
https://github.com/sonm-io/Contracts-scheme

Please also check here:
https://github.com/sonm-io/Factory/blob/master/contracts/Hubs/README.md

3. Container

The isolation system used as a core for computing by SONM, iEx and Golem is Docker. All development teams agree that this is the most up-to-date and useful solution.

4. P2P architecture

By using a hybrid P2P architecture, we can create computing pools out of the main number of miners/clients (similar to the cryptocurrency pools).Golem,while using full P2p, does not support the implementation of this case.

The specifics of SONM implementation.

The architecture has a **'hub'** which can serve as a regular client node (the client adds a number of tasks to the hub, which he created, and gives those to the miners), but also as **gateway** node, which can assemble mining power and perform tasks sent in by clients. That way the client doesn't need to implement his own node and fine-tune it to connect to the services. The services themselves will be discussed in detail in a further article.

5. SaaS

Software-As-A-Service implementation is the weak point of all grid systems. A large share of the cloud computing market is providing SaaS services (hosting websites, mobile apps, mail services, messengers, etc.). Currently SaaS is implemented and working in SONM, while Golem is planning to implement it in a year.

With regards TOR operations (The Onion Router), TOR uses pseudodomains.onion, and domain names look like the following: http://o3shuzjrnpzf2aiq.onion/

Domain names in the **.onion** domain are generated based on an open random key server and consist of 16 symbols. These websites are actually not websites at all, as they are in fact so-called hidden services. SONM is going to implement such services in the future, of which one application could be hosting websites. It is vital to take into account that storage and operation will be decentralized.

The realization will be in the form of free access to service data from the internet, or a process which is similar to the TOR system, which limits access. The structure of the service depends directly on the application running in the container.

In the address bar the service may look like the following: name,site.sonm or simply just %name%.sonm (which would be the name of the service that finds a hidden node using the locator and loads the website). This can be used for additional identification of services in the SONM network, granting them additional properties.

6. Fog computations

Cocaine has an integrated locator service, which allows the system to identify the service accessibility within the cluster. With a little fine-tuning our system can work as a true fog computing system – when a client requests access to a certain service, the locator (or a specifically created additional global higher-layer locator) will identify the cluster closest to him and perform the computations using it, thereby sufficiently reducing traffic.

Speaking about iEx and Golem, they have yet to mention the implementation of this service.

To summarize:

It is currently hard to describe Golem as a serious project. We call the current state of affairs reinventing the wheel. therefore, only a serious change in the direction of their work can permit them to stay in the field.

SONM, on the other hand, is using the Distributed Cloud technology as its base. The SONM team is modifying said technology to fit our current goals in producing a less work-intensive endeavor than combining Xtermweb-hep (base used by iEx) with traditional Cloud services, due to Grid software being severely limited in that regard, and developing it to commercial standards is very difficult and time-consuming. This is the reason SONM decided not to use BOINC, which is one of the Grid software solutions, similar to Xtremweb-hep.

Having a prototype system for verifying undetermined computations also gives a hefty advantage to SONM.

Implementation time and flexibility in technologies used will be the deciding factor at the starting phase of the project and will give us a continuous edge.

SONM is now looking to hire world-class specialists on P2P networks and locators (for optimization) and people with experience working with BTSync and Tor Browser software.

5. REFERENCES

[1] https://en.wikipedia.org/wiki/Fog_computing

[2] https://en.wikipedia.org/wiki/Internet_of_things

[3] http://internetofeverything.cisco.com/sites/default/files/docs/en/ioe_value_at_stake_public_sector%20_analysis_faq_121913final.pdf

[4] https://en.wikipedia.org/wiki/Crypto-anarchism

[5] http://internetofthingsagenda.techtarget.com/definition/fog-computing-fogging

[6] https://en.wikipedia.org/wiki/Turing_machine

[7] https://users.soe.ucsc.edu/~abadi/Papers/isss02.pdf

[8] https://www.cs.bham.ac.uk/~mdr/research/papers/pdf/11-applied-pi.extended.pdf

[9] Verifying computations without reexecuting them: from theoretical possibility to near practicality. Walfish, Blumberg.

[10] Making Argument Systems for Outsourced Computation Practical (Sometimes). Setty, McPherson, Blumberg, Walfish.

[11] Practical Delegation of Computation using Multiple Servers. Canetti, Riva, Rothblum.

[12] An Intro to TrueBit: A Scalable, Decentralized Computational Court. Simon de la Rouviere.

[13] https://en.wikipedia.org/wiki/Combinatorial_optimization

[14] https://en.wikipedia.org/wiki/Knapsack_problem

[15] https://en.wikipedia.org/wiki/Travelling_salesman_problem

SONM

FOLLOW US AND STAY TUNED

sonm.io

Slack

Reddit

Telegram

Twitter

BitcoinTalk

Medium

Facebook

GitHub