# Decentralized Network of Services

Migrating traditional technologies towards decentralization

Anthony Estebe
anthony@mesg.com

Nicolas Mahé
nicolas@mesg.com

November 2018
https://mesg.com

## 1 Introduction

The web has evolved to a point where traditional web technologies are now co-existing alongside newer, secure and interactive technologies such as blockchains and machine learning engines. These new technologies have vast improvements over the traditional web, yet both old and new still serve a purpose, therefore must run in parallel to each other.
Creating applications which migrate traditional web services towards trustless, interactive services is currently costly and difficult, and has many inherent limitations. While web service ecosystems are helpful, they are difficult to use and functionality is limited because they cannot connect to each other.
The world needs a way to connect web service ecosystems to new systems like blockchains, while maintaining the spirit of the newer trustless, decentralized solutions. To allow both to exist harmoniously, a way to create applications on top of a decentralized infrastructure which supports the bridging to traditional web service ecosystems is needed.

## 2 Network of Services

After many years of software and web development, certain concepts have proven to be more efficient than others. Service Oriented Architectures and Event Driven Architecture have become widely adopted because they are powerful and easy to scale. This inherent scalability also makes these architectures easy to decentralize.

### 2.1 Service design

Services are a way to connect to specific business logic, a feature or a technology. They typically feature bi-directional communication. Services receive task to execute (like most classic Service Oriented Frameworks) and proactively send events related to their business logic.

Any incoming data on a service is categorized as a task, and any outgoing data is categorized as an event.

### 2.2 Trustless services

To create a trustless environment, services need to be validatable.
Blockchains have adopted deterministic virtual machines to validate executions by allowing other actors on the network to execute them again to check that the result matches. This is not the only way executions can be validated.
In fact, most traditional web technologies are already validatable. For example, one actor of the network executes a task that records data in a database and other actors validate it by checking that the data is still present and has remained unchanged.
In order to have a fully-reliable trustless system, services' tasks and events need to be validated by a network of services that requires validation by many different actors on the network.

## 3 Verification consensus

Each service is different (services can be connected to blockchains, IoT devices, centralized databases, etc.) and requires a different level of verification depending on applications' requirements. This extent of verification can range from no verification to verified by all nodes, meaning they can range from a fully-trusted service to a fully-trustless service.
This network of services should allow a flexible level of trust. As long as a service is verifiable, the decision to use this service in a more or less trustless and decentralized way belongs to applications.
In specific use cases, a really high level of verification and decentralization is not necessary and can considerably slow down applications. With this approach, applications can determine how many actors should be involved in the validation of the service. Of course, if the service is a node that is trusted by applications, the number of verifications can be set to zero. On the other hand, if the node is not known (in the case of full decentralization) applications might want a high number of validations to make sure that the service is executed upon correctly.
This level of trust is applied to the two components of services:

- Events emitted by the services
- Tasks executed by the services

### 3.1 Event consensus

Events are observed and emitted by services, but every node running a service can willingly or unintentionally emit wrong or inaccurate events. In order to prevent this, the network ensures that events are valid.

Once an event occurs, a node hashes the data of the event and then shares it with the rest of the nodes based on an unique deterministic event ID that the service provides.
When a majority of nodes observe the same event within a specific period of time (which can be defined within the service depending of the estimated frequency of this event) the event is marked as valid.

### 3.2 Task consensus

Task executions need to be verified as well. This level of verification differs for every execution, based on the level of decentralization needed by applications. The verification mechanism has dynamic parameters but always involves two types of actors: executor and validators.
Task executions are handled by a single executor and multiple validators. The executor is the node executing the task. Validators are the nodes validating the execution, either by reproducing exactly the same task and comparing the outputs (deterministic task) or by executing a verification method (non-deterministic task, but still verifiable).

## 4 Stability of the network

In order to make sure that the network is reliable and that enough actors are participating in the validation and execution process, the network needs to provide incentives for the nodes that are helping the network.
These incentives need to be favourable for the good actors on the network, but also be extremely costly for those who show malicious behavior.

### 4.1 Staking system

These behaviors only concern nodes that are participating in the relevant service. In order to participate in the service's execution or validation, every node needs to provide a commitment that they will behave correctly and will continue running this service for a minimum period of time. To fill the contract, nodes need to stake tokens to attest to their good intentions.
Once this contract is digitally signed, any malicious behavior detected will result in the loss of a part of the stake to ensure that nodes behave correctly.
The system needs to be able to detect malicious behavior which can originate from the three different actors:

- Emitters that submit incorrect events
- Executor that execute a task incorrectly
- Validators that validate a task's execution incorrectly

### 4.2 Detecting malicious behavior

The portion of the stake lost is relative to the severity of the malicious behavior and the number of nodes involved in the process (event emission, task execution or task validation).
The fewer nodes involved in the process, the higher the penalty is for malicious behavior.
The more nodes involved in the process, the lower the penalty is for malicious behavior.
Events are checked by the consensus mechanism (see "Event Consensus"). Every node that is running a service will submit their version of the events with all necessary information. If a node emits a result that differs from the reached consensus, then the node's stake is slashed.
Tasks are checked by the validators of this task. If the result submitted by the executor is proven invalid by validators, then the executor's stake is slashed. Tasks are critical because they are executed by a single selected node in the network, so each case of malicious behavior needs to result in a strong penalty.
Validators are the last step and they need to be regulated as well. This regulation is done in the same way as the events: if the verification of a validator is not in the majority, then it is considered an invalid verification and a portion of their staked tokens are slashed. This part is critical, and needs to be as secure as possible by ensuring that malicious behavior is very costly for the misbehaving nodes.

### 4.3 Incentivize good actors

In order to have nodes absorbing the risk and to make sure that they do their work correctly, an incentive system needs to be implemented to motivate them to correctly do the job they are supposed to do. If a node provides a good event, task or validation, the node will receive tokens for its work in supporting the network. This incentive should be large enough to encourage users to run nodes, and the penalty should also be large enough to discourage them from behaving maliciously.
A reputation is built based on behavior and actors with a good reputation take more importance in the network.
Applications pay fees to use the network. These fees are distributed directly to the different actors who contribute to the application's use.
The network rewards good behavior by creating new tokens that are automatically distributed to the actors of the network based on their reputation and involvement.

## 5 Node selection

The three different kind of actors (emitters, executors, validators) need to be selected wisely. If a node is selected but this node misbehaves, it will have a big

impact on the application. To make sure that only the best nodes are selected, this selection should be based on different criteria.

- The service that a node is running
- The amount of tokens staked for this service
- The time period of committed support to this service
- The previous successful emission/execution/validation rate based on the requested task
- The total amount of stake previously taken / number of incidents of malicious behavior
- The time period since the last behavior
- The availability/uptime of the node

By combining all these criteria, the network can select a node for executing a task. The selection needs to be deterministic to ensure that every emitter agrees on the same executor and validators to handle the execution. Additional criteria can be added to ensure a wide actor selection. Some "randomness" should also be included, based on the data provided by the event, to not always select necessarily the best nodes and to give opportunity to other nodes. This ensures that the nodes can be selected according to the data they will process.

## 5.1 Custom distribution

Users should have the opportunity to manually select a set of nodes. This gives users the possibility of creating sub-networks that could be used for running trusted executions.
Traditional service providers can progressively migrate towards trustless decentralization by first running their services in a trusted network, then adding verifications (the trust component), and then finally allowing anyone to execute their services by opening their network (the decentralized component).

## 5.2 Data security

Because all actors are known and selected, data can be encrypted specifically for them. This ensures that confidential data can't be revealed to entities that shouldn't have access to it. It can be combined with a custom node selection, allowing for full privacy on the data used.

# 6 Conclusion

This network of services solves the issue of migrating traditional web technologies to trustless and decentralized systems by also giving them the possibility to progressively migrate.
This migration can be done in the following steps:

- Allowing the service to be accessible from a decentralized network while remaining trustful (trusted executions)
- Removing the trust from services by providing task verification functionality
- Allowing other nodes to run services to have a fully-trustable and decentralized service.

It also provides a single method of communication for any kind of service, making traditional web technologies trustless and decentralized technologies able to communicate with each other (and any other technologies like IoT).
Service executions can now be monetized and traditional web services can have a new way of monetizing their technologies.
It allows the creation of applications based on event-driven architecture that connect events to tasks, with verified and consensus at every step. This opens up the possibility of creating real decentralized applications that can choose their level of decentralization, according to the need of users.