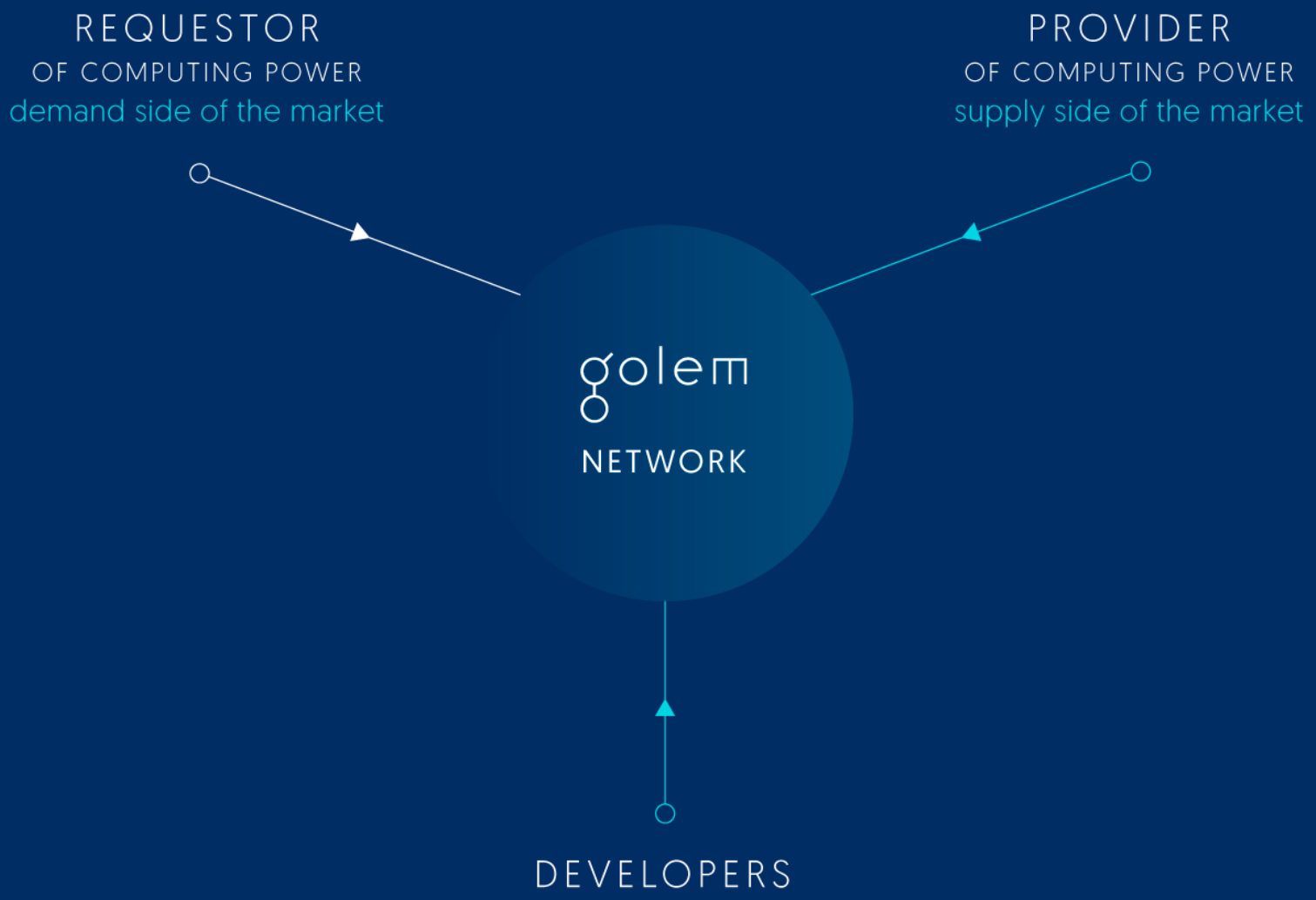
WORLDWIDE
SUPERCOMPUTER

BY
PIOTR JANIUK

GOLEM: THREE SIDED MARKET



CHALLENGES OF DECENTRALIZED SETTING

01

Game-theoretical standpoint for all actors

02

Trust in the network

03

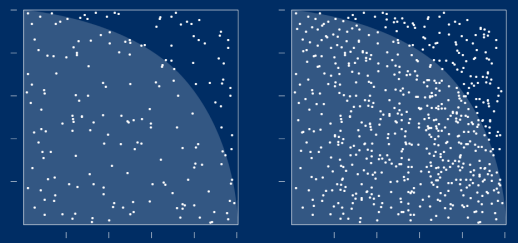
Strict verification requirement

04

Nondeterministic computations

TrueBit

https://truebit.io - can be used in case of deterministic computations



Monte Carlo methods

WHY RENDERING?

THE DOMAIN OF THE PROBLEM IS WELL KNOWN

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_\Omega \rho(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') \cos\theta d\vec{\omega}'$$
$$\cos\theta = (\vec{\omega}' \cdot \vec{n})$$

CG IS PRESENT IN MOST MEDIA, AND PLEASING RESULTS ARE ACHIEVED FAIRLY QUICKLY

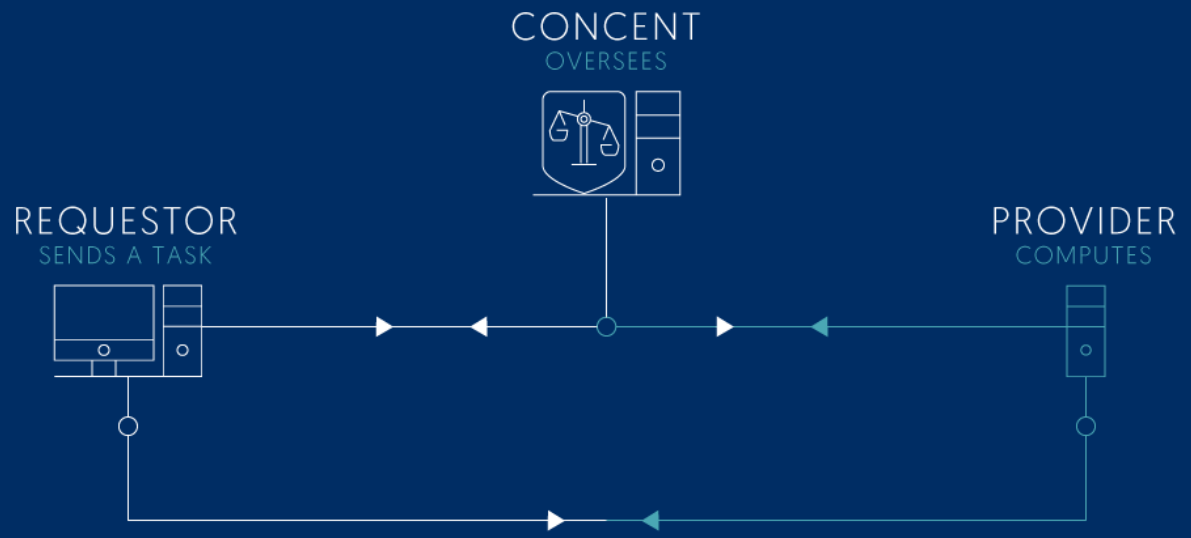
source: https://www.behance.net/gallery/21615831/Testing-render-engines

TRUST AND SECURITY

01

Concent

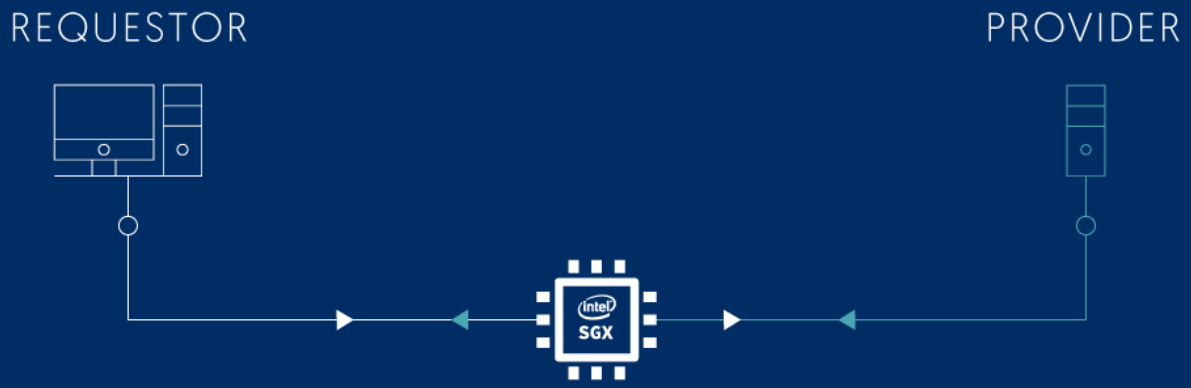
- Is a service in the Golem network, which aims to improve the integrity and security of transactions



02

Intel SGX technology

- Technology which can be used to implement secure remote computation



A BRIEF INTRO TO GOLEM TECH

GOLEM TECHNOLOGY
and Nondeterminism

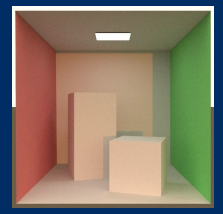
FIRST POC

Focus on computations / computation-centric use case

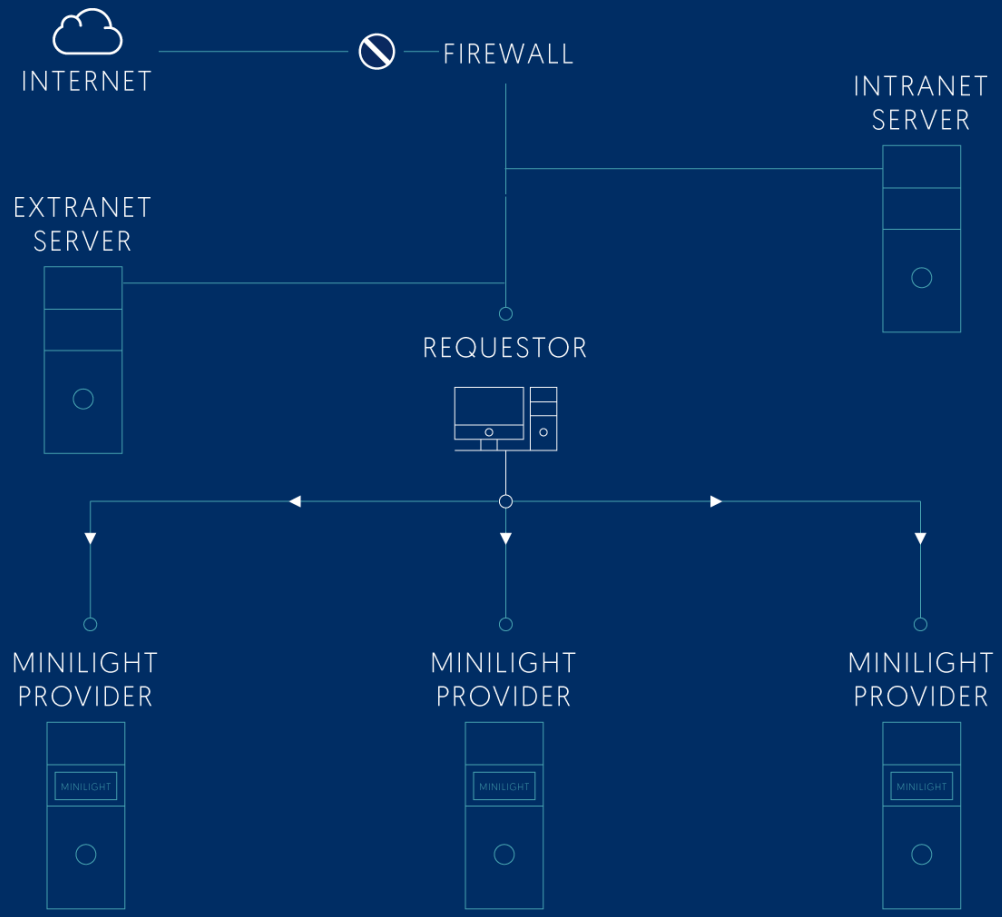
01

Specific renderer hardcoded in provider

(POC 0.5 - Minilight)



Minilight (Python)



FIRST POC

Focus on computations / computation-centric use case

02

Rendering as a separate task

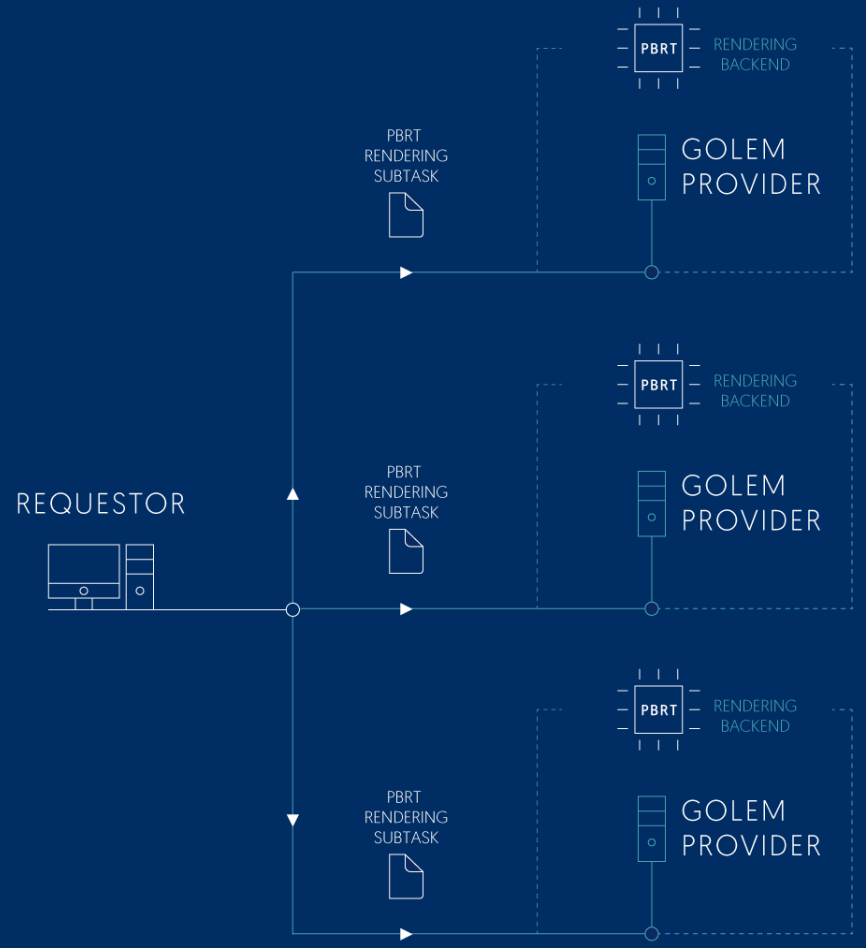
- Rendering task is abstracted out and decoupled from the Golem client code. The first implementation used PBRT renderer as a rendering backend

pbrt (C++)

03

Basic protocol

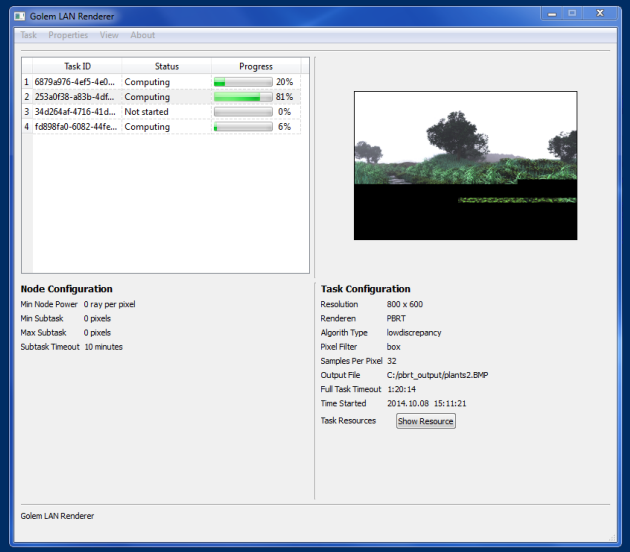
- Trusted computing nodes
- Lack of transaction framework
- No payments

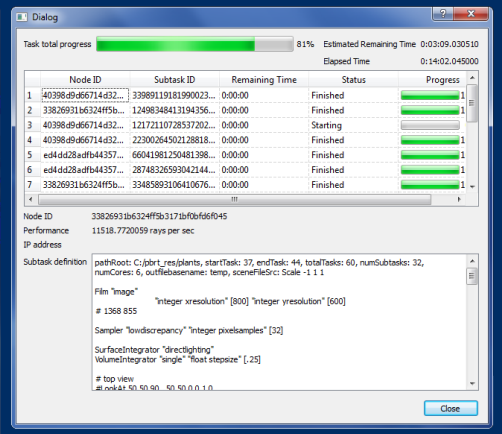


FIRST POC

01

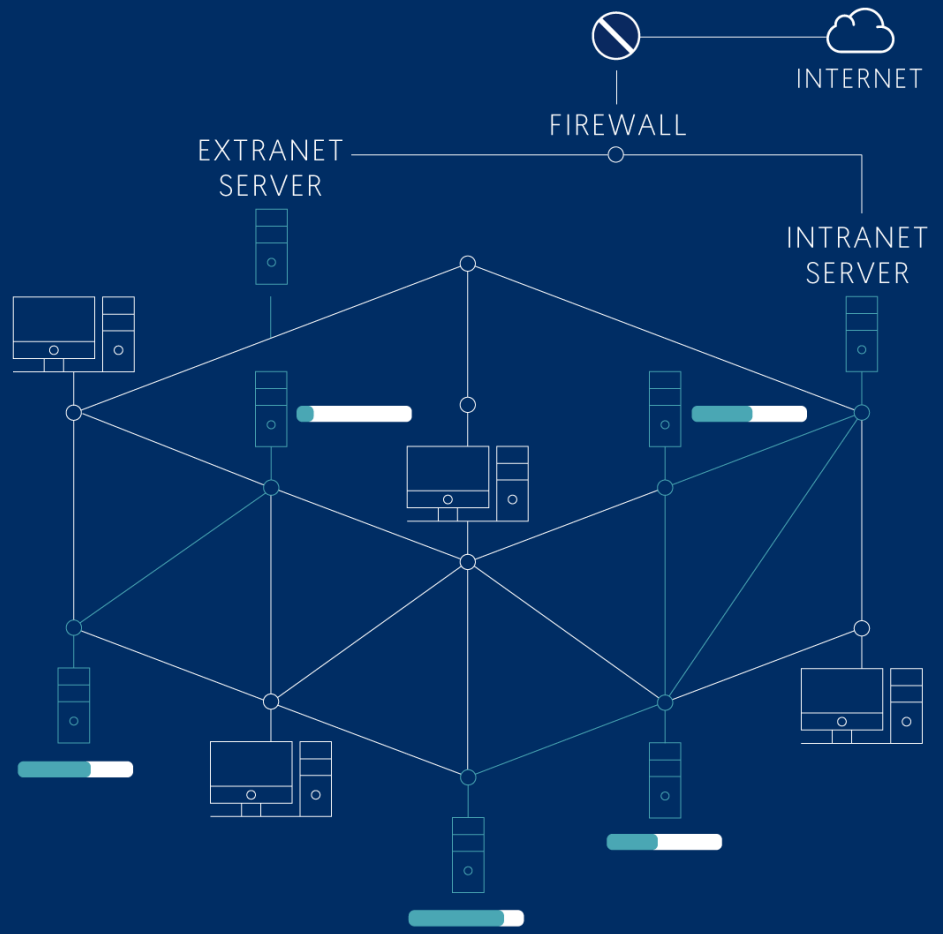
First POC





02

Intranet computations



TASK POV

01

High level task definition

02

Task configuration:

- Timeouts (global, local)
- Task splitting configuration

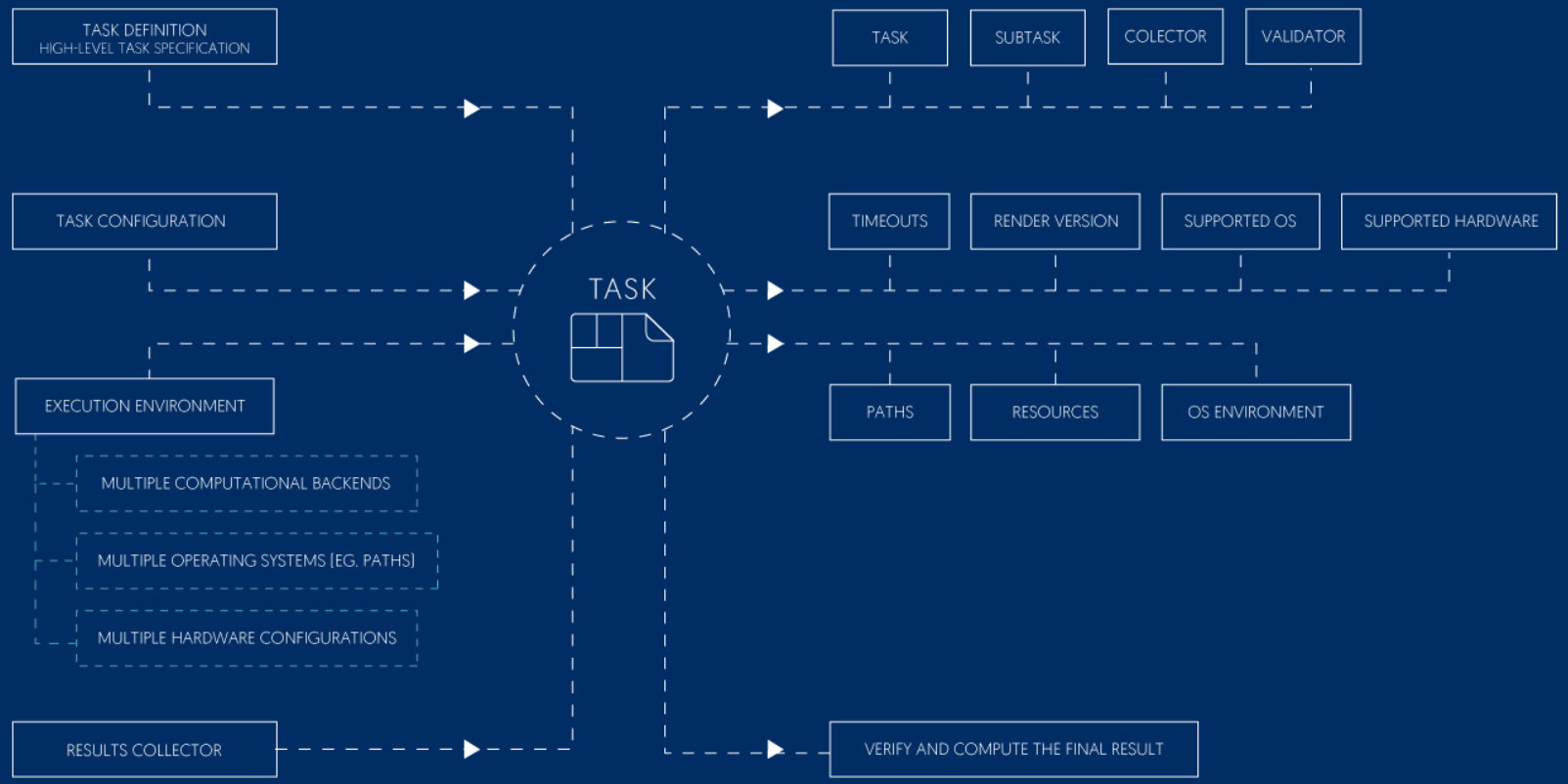
03

Execution environment:

- Multiple computational backends
- Multiple operating systems
- Multiple hardware configurations

04

Subtask results collector



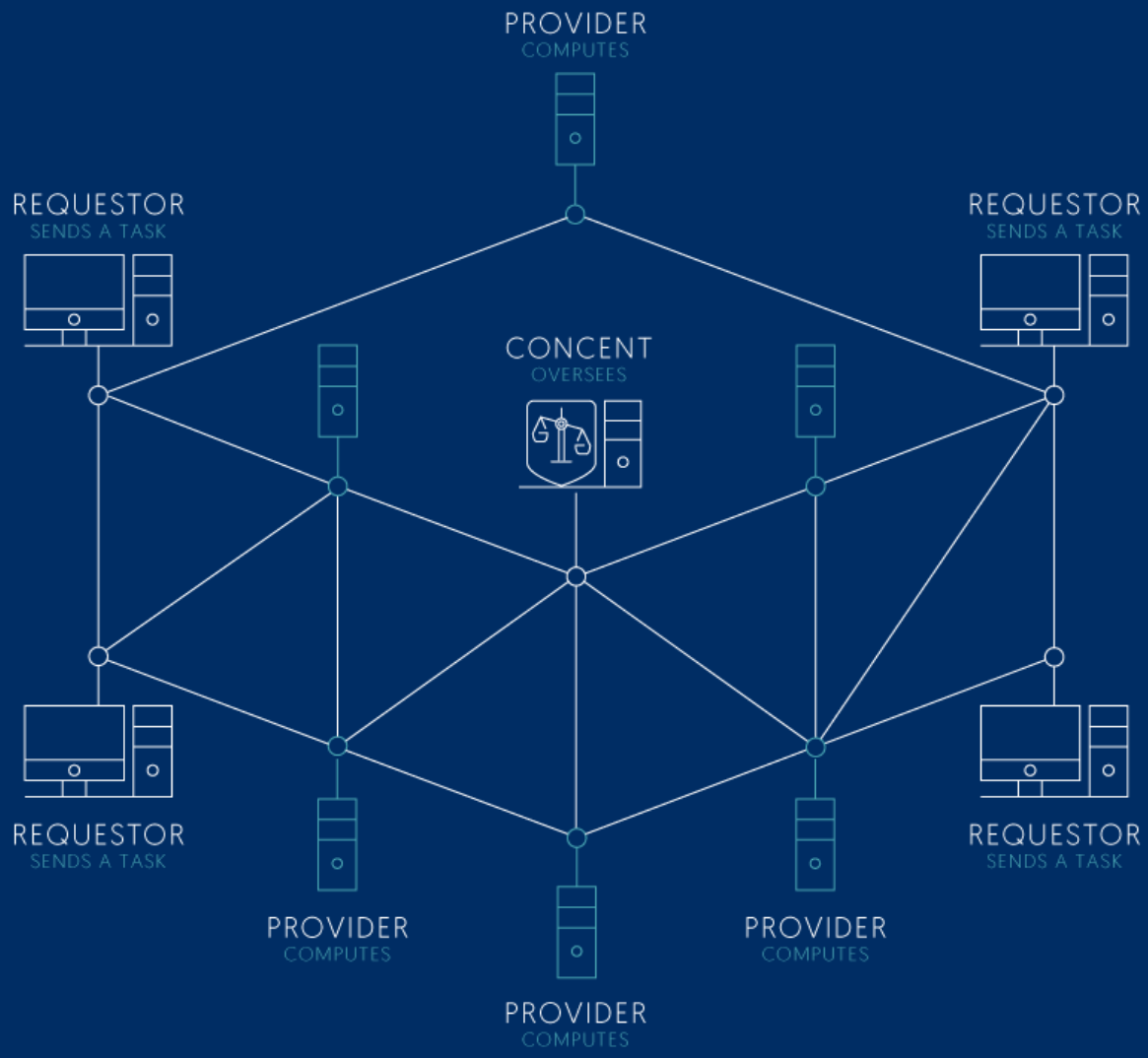
COMPUTATION ENVIRONMENT

01

Physical infrastructure:
Heterogenous p2p network

02

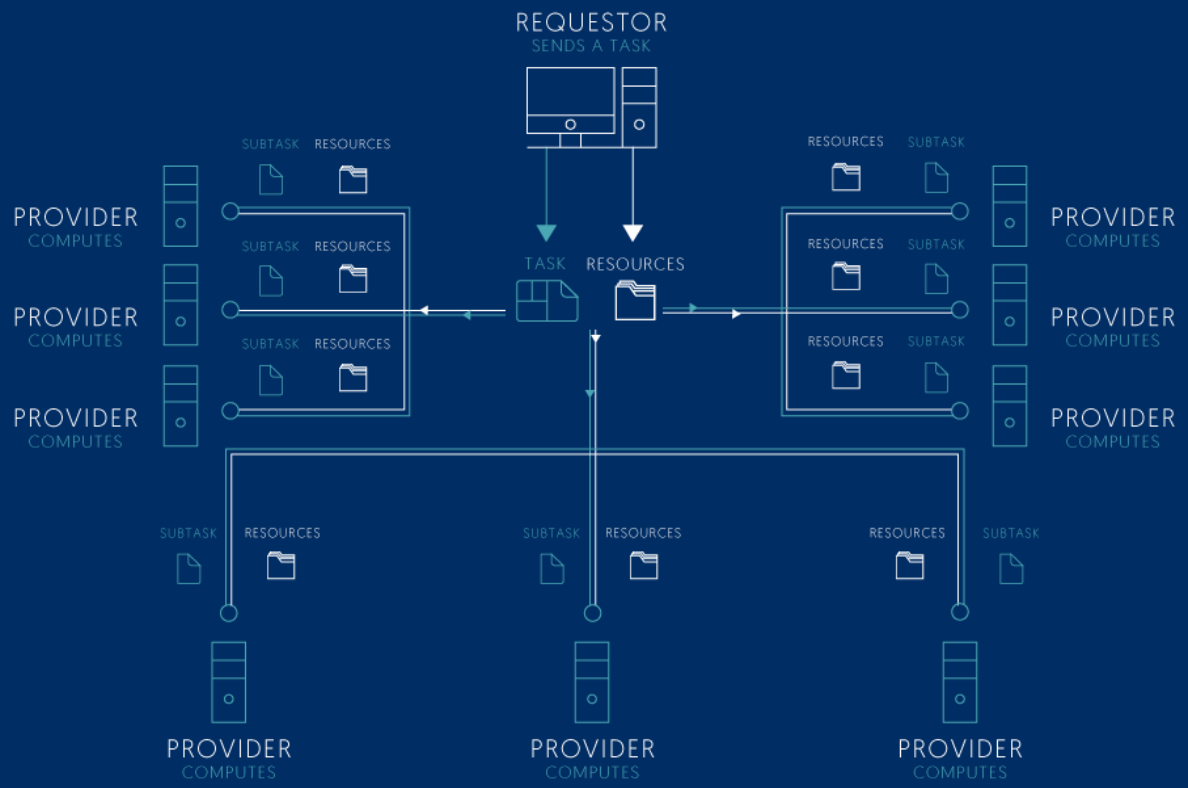
Logical components: Requestors, providers, [optional] concents



COMPUTATION ENVIRONMENT

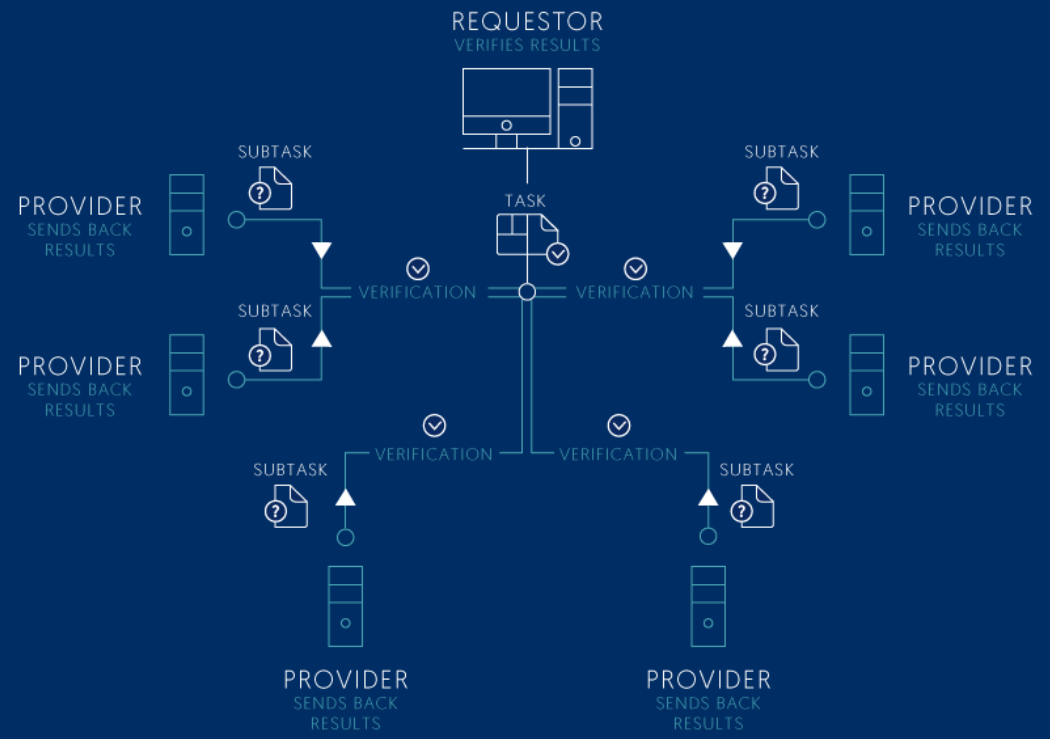
03

Task sending and resource distribution



04

Verification



SECURITY CONSIDERATIONS

01

Provider security

- Isolating host from possibly malicious software
- Accepting only valid tasks

02

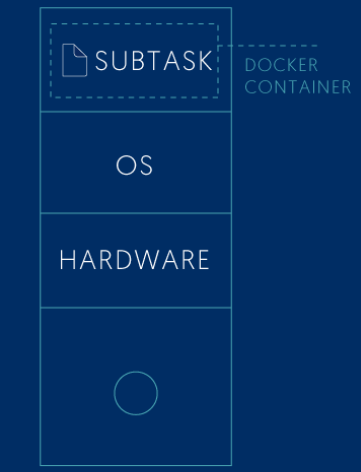
Requestor security

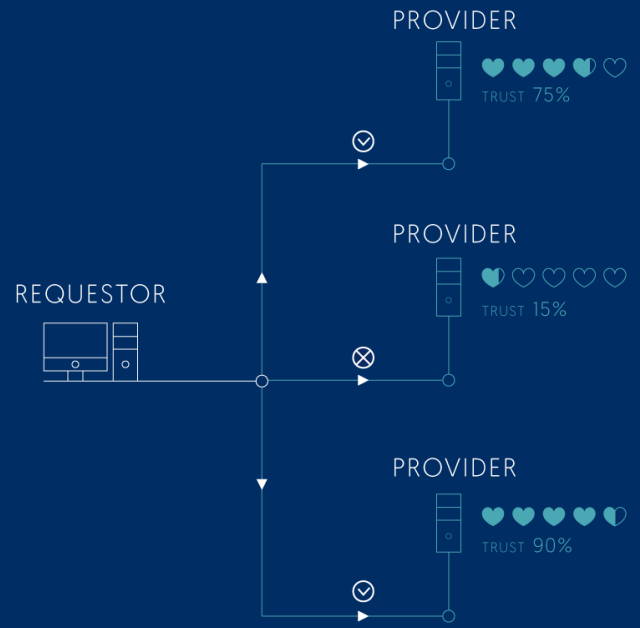
- Trust (i.e. reputation)
- Results verification
- Keep track of providers and notion of trust

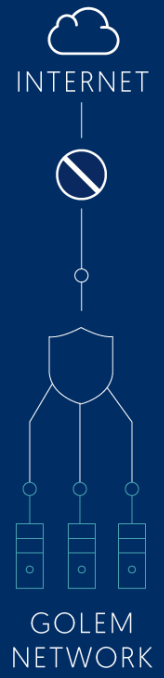
03

Infrastructure security

- no unauthorized internet access (e.g. botnet protection)







INFRASTRUCTURE AND ACTORS

01

New topology and requirements – P2P

02

Distribution of execution environments

03

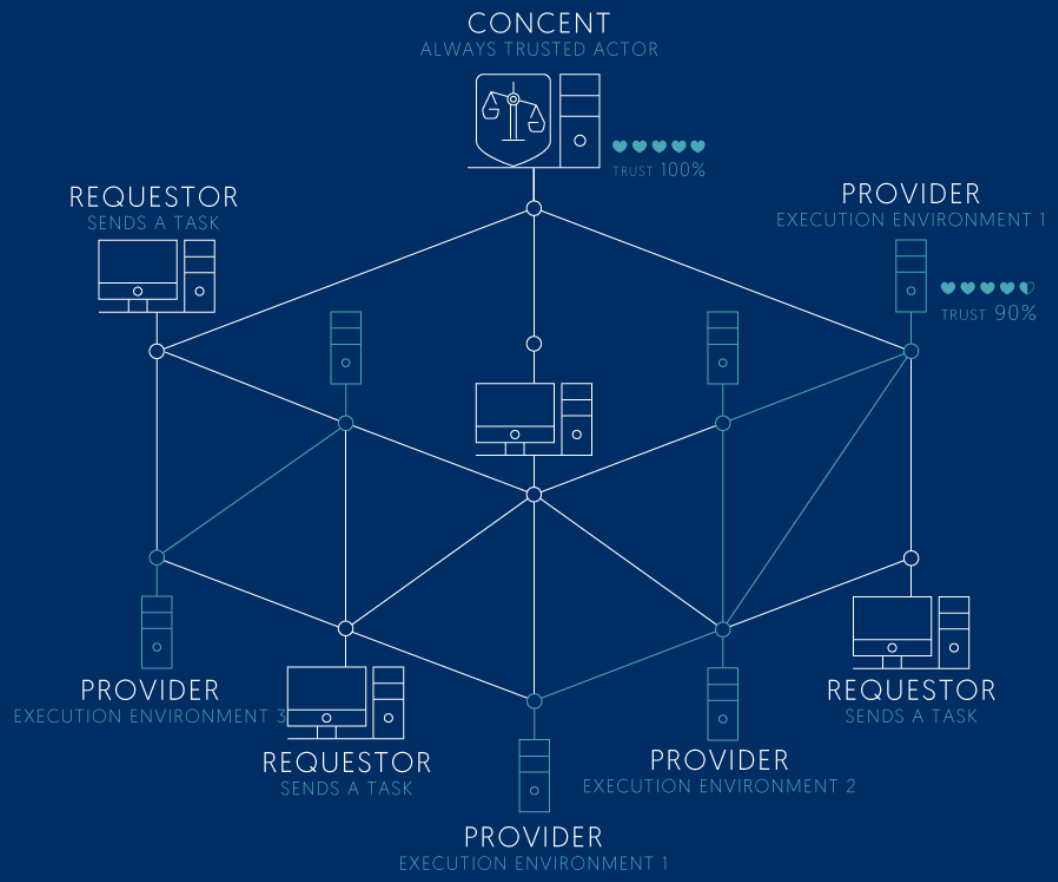
Notion of trust – reputation of the actors

04

New integrations – new use-cases

05

Actors in the network



INCENTIVES AND ECONOMY

01

Game-theoretical conditions for all actors

- Notion of trust in the network (local and global schemes)
- Known verification requirements and algorithms
- Open market

02

Requestor incentives

- Wants to compute cheap, fast and get a high-quality result

03

Provider incentives

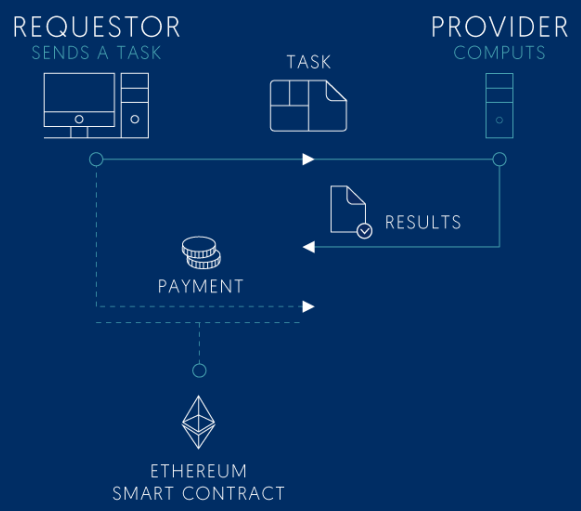
- Wants to compute fast and make the highest possible profit

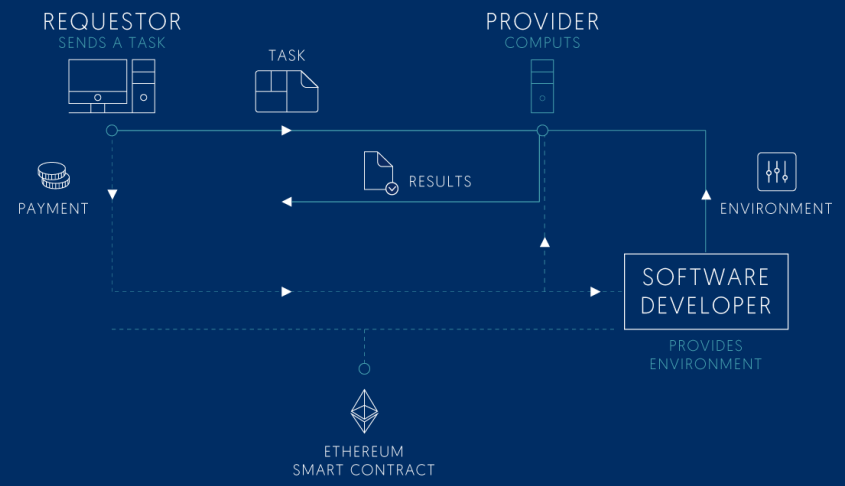
PAYMENTS INSIDE THE NETWORK

01

Reward scheme:

- Every integration can use any custom reward scheme, as long as it is compliant with the transaction framework
- Key parties are:
  - task creators (Requestors): how they pay for computations and software
  - Providers: how they are rewarded (eg. per hour? per task?)
  - Developers: payment for software (free? donation? per node? per use? per CPU hour
- Ethereum network is used to transfer value and information between the peers





A BRIEF INTRO TO GOLEM TECH

Golem Technology
AND **NONDETERMINISM**

FIRST USE-CASE (MC PHOTOREALISTIC RENDERING)

SOURCES OF NONDETERMINISM

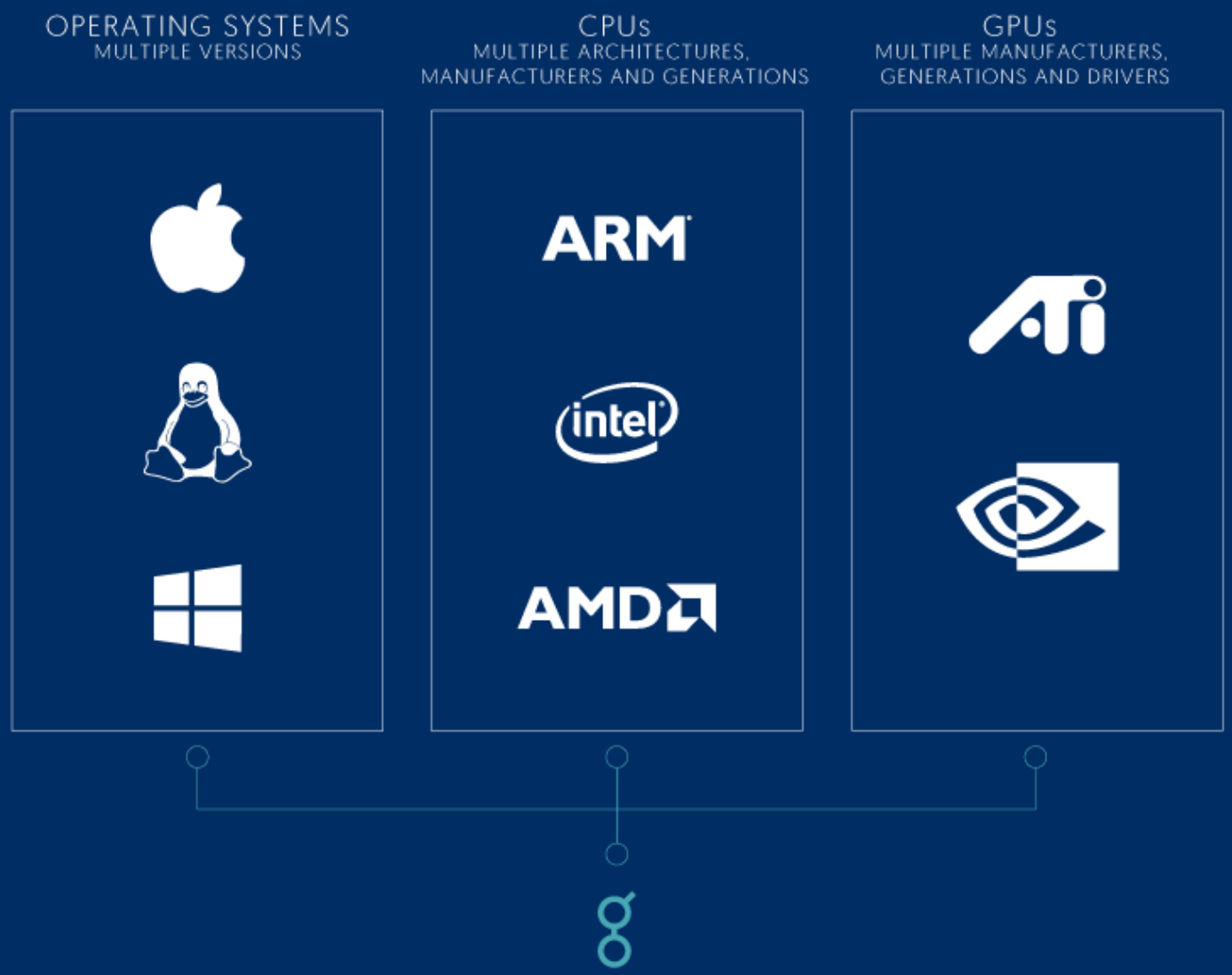
01

Floating point encoding standards on different platforms:

- CPUs
- GPUs

02

Random number generators



SOURCES OF NONDETERMINISM

01

Float encoding standards on different platforms:

- CPUs
- GPUs

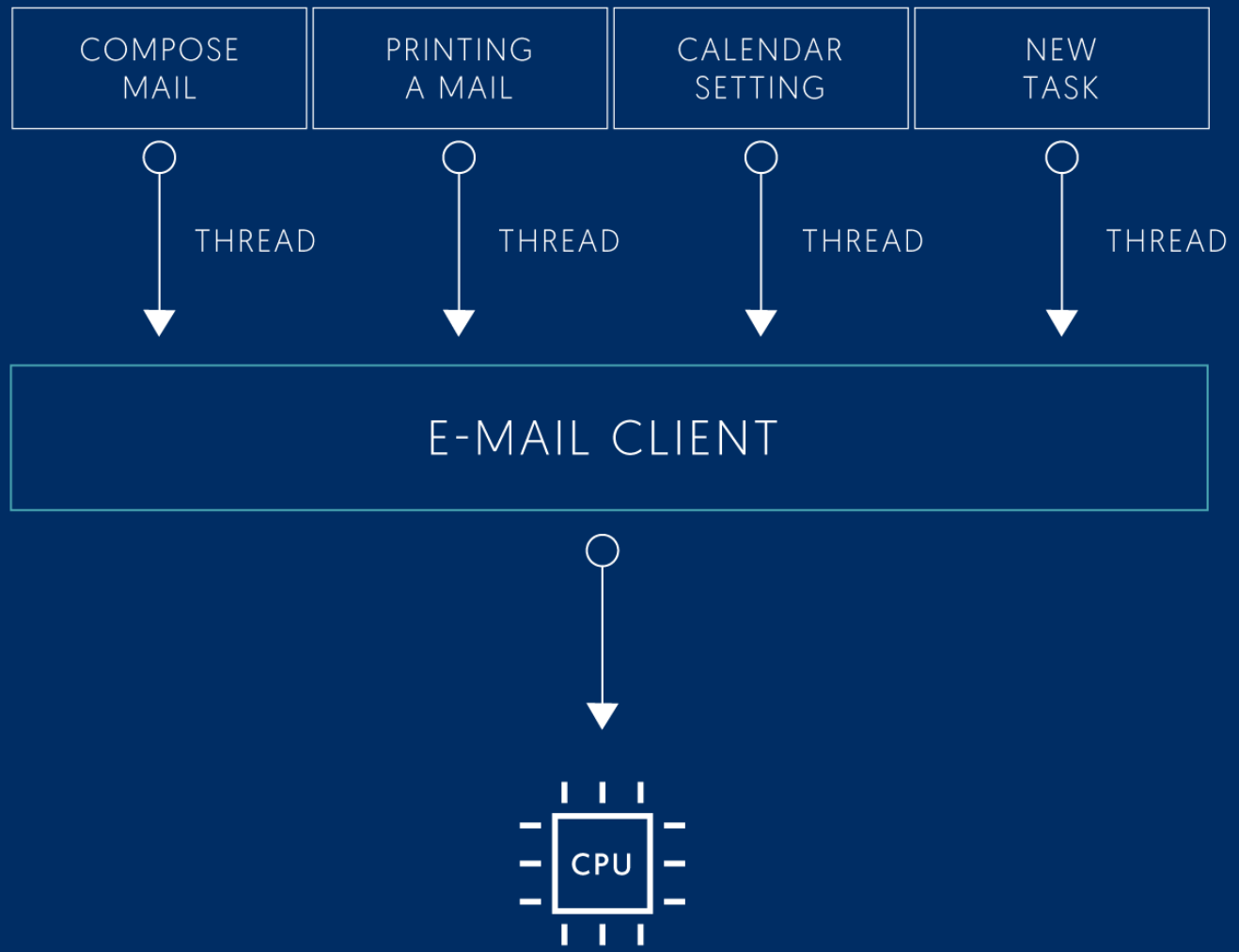
02

Random number generators

03

Multithreading:

- Regular CPU multithreading



SOURCES OF NONDETERMINISM

01

Float encoding standards on different platforms:

- CPUs
- GPUs

02

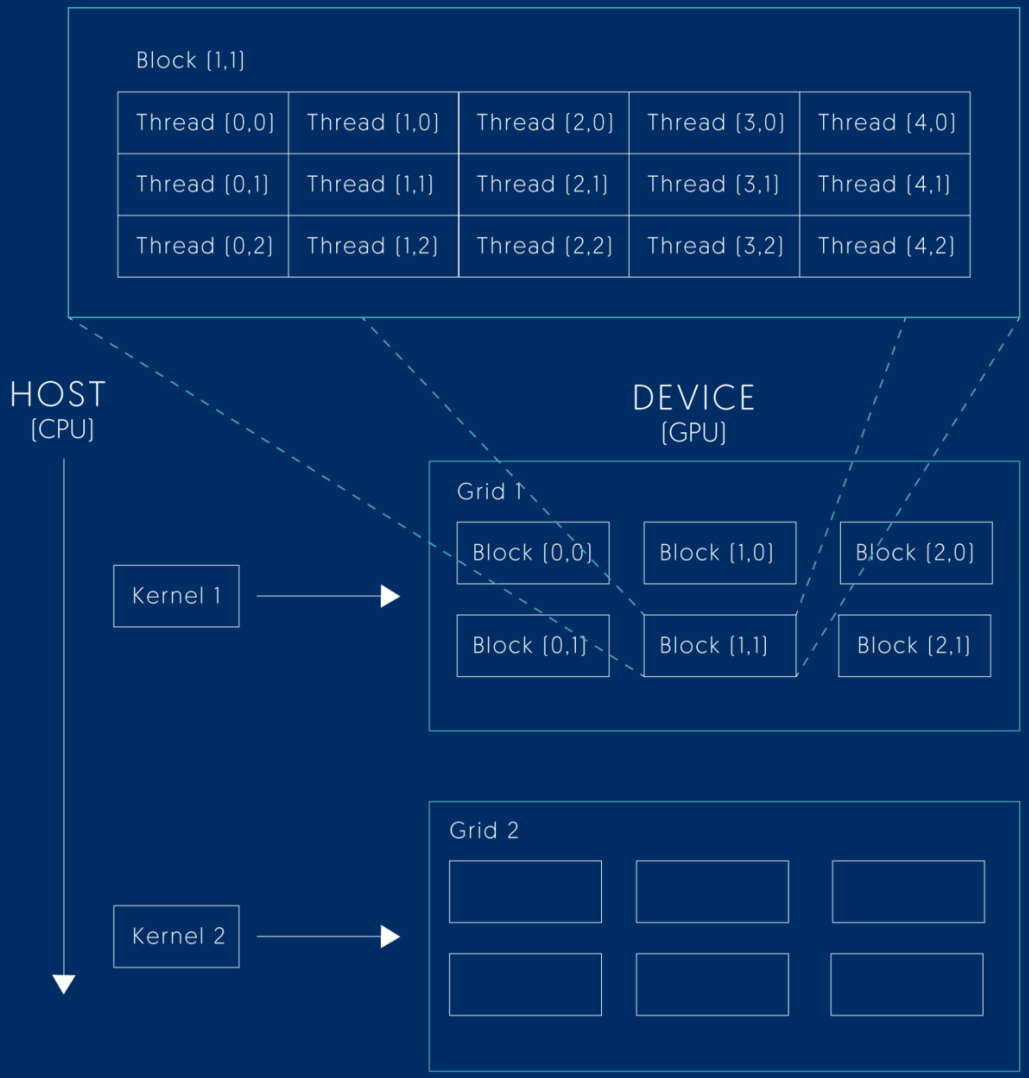
Random number generators

03

Multithreading:

- Regular CPU multithreading
- SIMT or SMT in the case of GPUs

CUDA ARCHITECTURE



SOURCES OF NONDETERMINISM

01

Float encoding standards on different platforms:

- CPUs
- GPUs

02

Random number generators

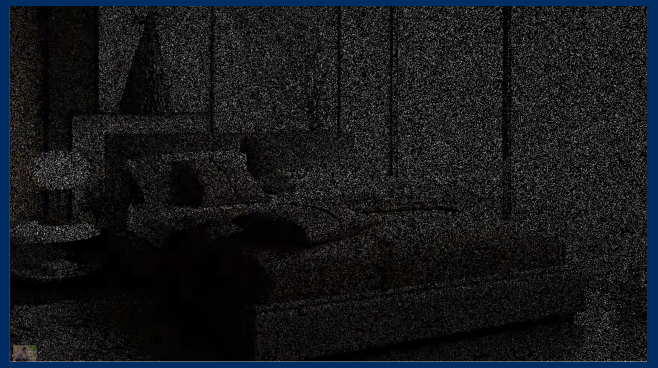
03

Multithreading:

- Regular CPU multithreading
- SIMT or SMT in the case of GPUs

04

Nondeterministic implementation of MC methods

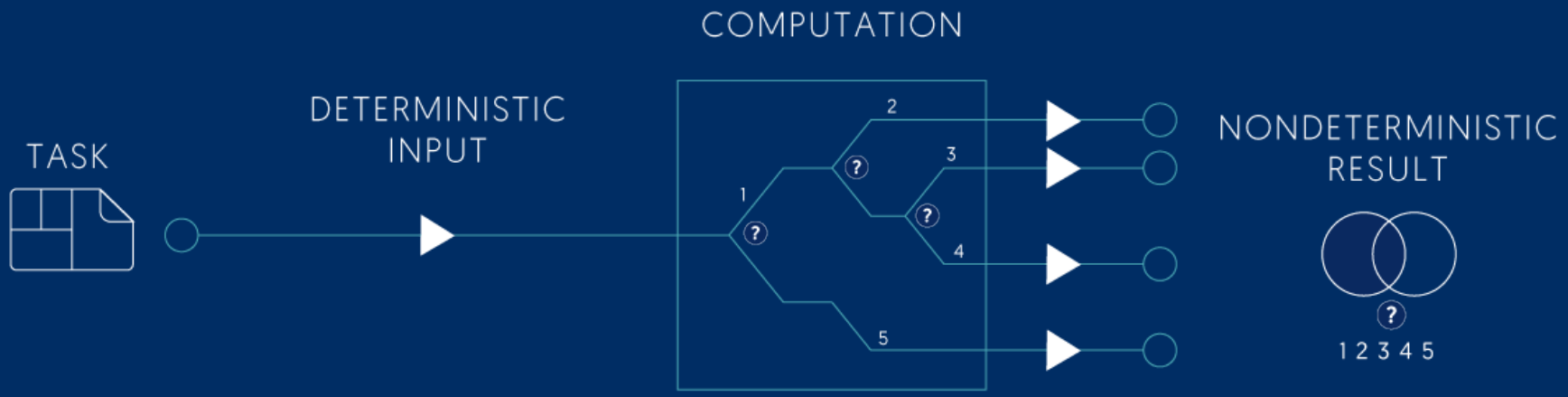


PROBLEMS WITH NONDETERMINISM

01

Rendering

- Deterministic input
- Nondeterministic computation
- Nondeterministic results

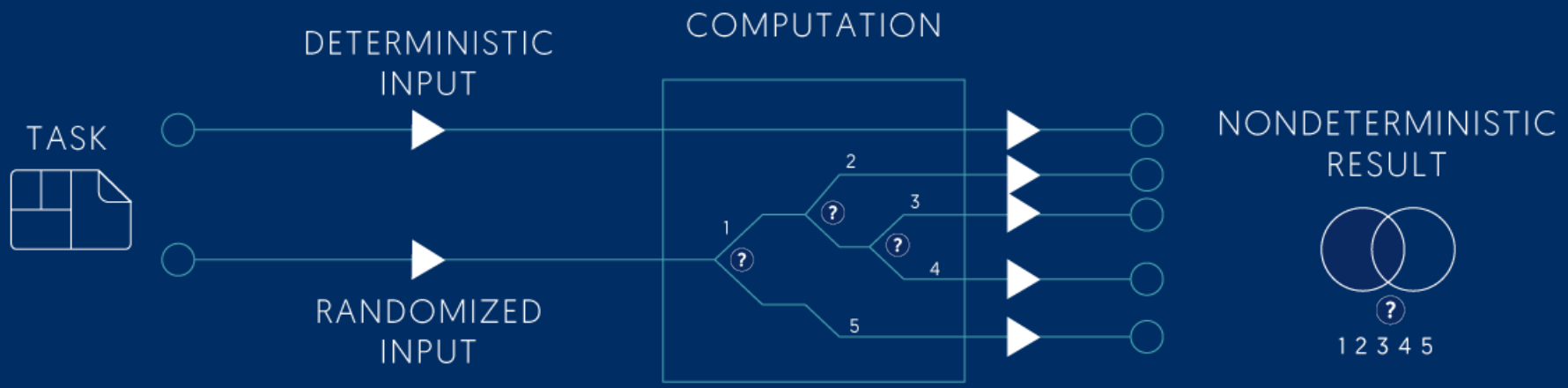


PROBLEMS WITH NONDETERMINISM

02

Machine Learning

- Deterministic or randomized input
- Deterministic or nondeterministic computations
- Nondeterministic results

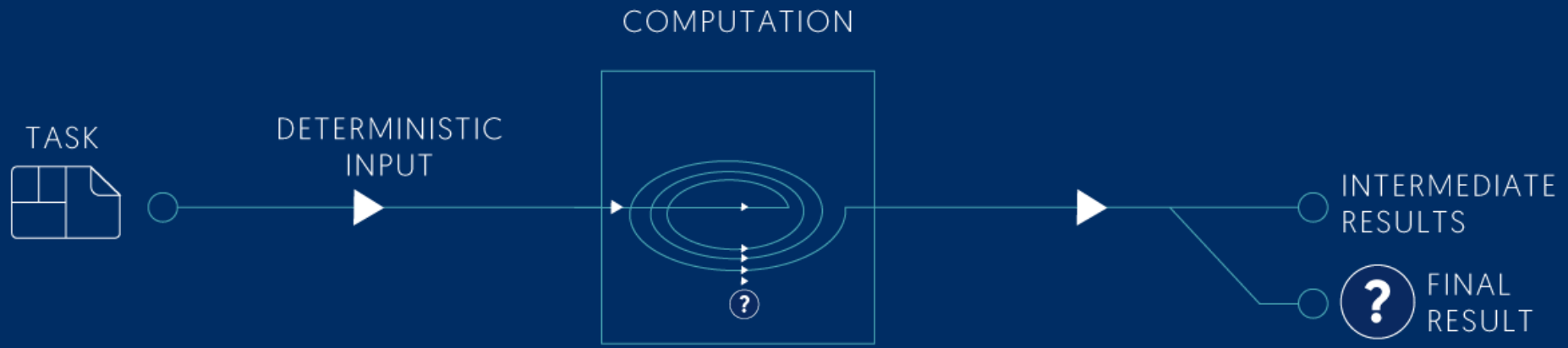


PROBLEMS WITH NONDETERMINISM

03

Proof of Work

- Deterministic input
- Nondeterministic amount of work required
- Deterministic and efficient verification
- The exact amount of performed computations is hard to verify

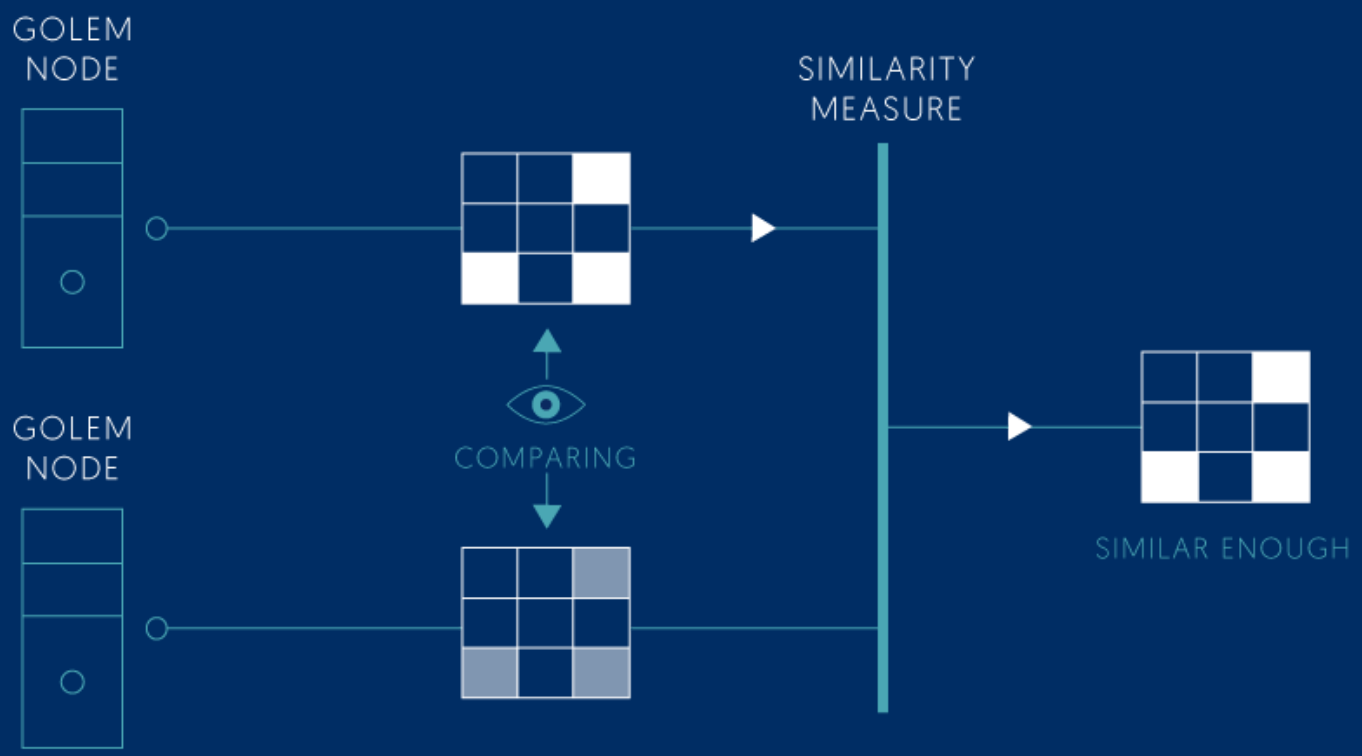


NONDETERMINISM AND CONSENSUS

01

Nondeterministic state

- Nondeterministic results cannot be directly compared
- Each node may see a slightly different result
- A similarity measure is required

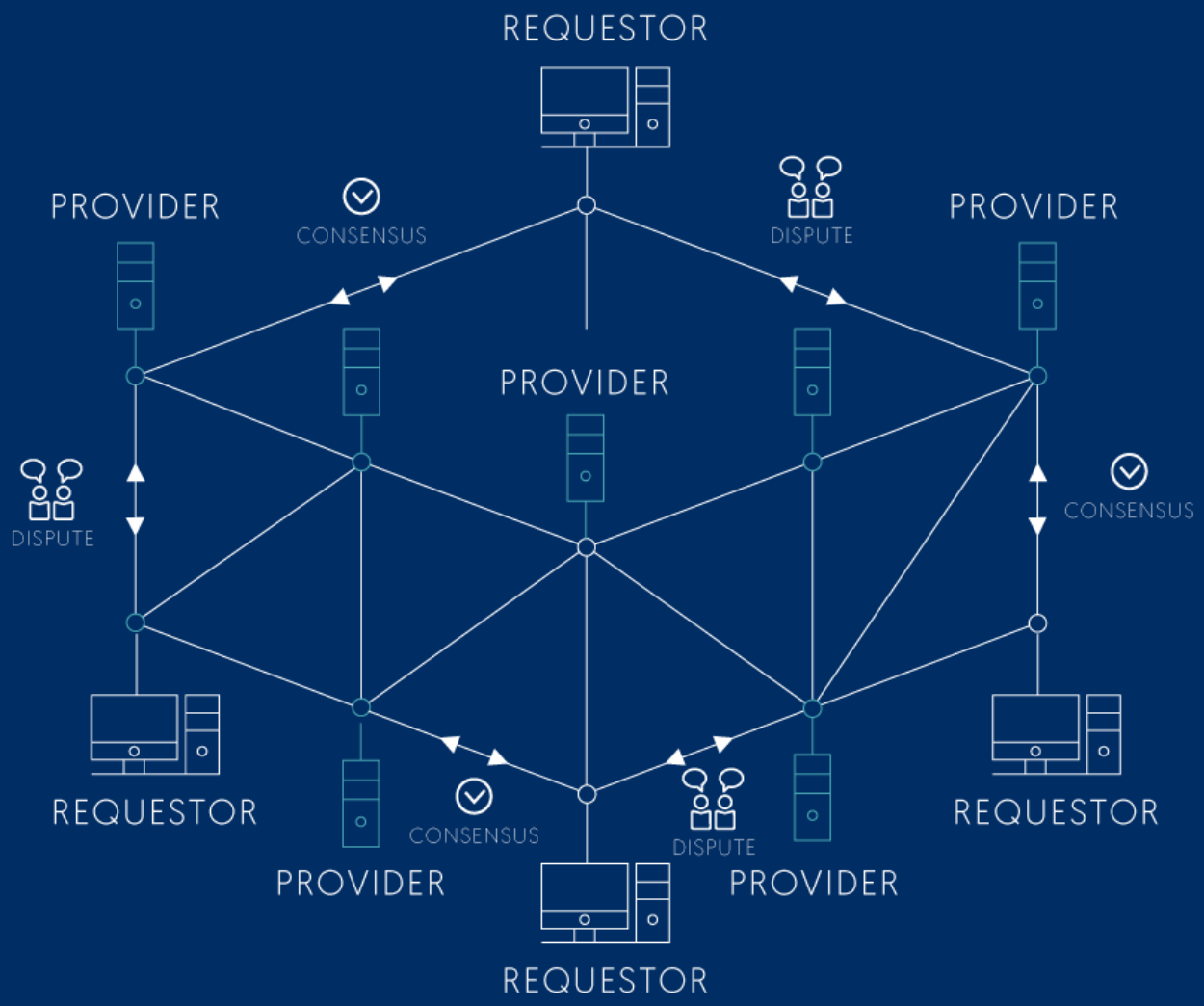


NONDETERMINISM AND CONSENSUS

02

Golem's setting

- Consensus between a very small (usually two) number of nodes
- Means of dealing with a dispute are implemented



MONTE CARLO METHODS

Introduction

BASIC PROBABILITY

01

Probabilistic space

PROBABILISTIC SPACE

Ω – domain

dP – probabilistic measure

$$\int_\Omega dP(x) = 1$$

02

Probabilistic measure via density function

PROBABILITY DENSITY FUNCTION

$$p(x) \geq 0$$

$$dP(x) = p(x)dx$$

$$\int_\Omega dP(x) = \int_\Omega p(x)dx = 1$$

BASIC PROBABILITY

> A random variable, usually written X, is a variable whose possible values are numerical outcomes of a random phenomenon. There are two types of random variables, discrete and continuous.
>
> – VALERIE J. EASTON AND JOHN H. MCCOLL'S STATISTICS GLOSSARY V1.1

03

Random variable

RANDOM VARIABLE

$$P(x \in S) = \int_S dP(x) = \int_S p(x)dx$$

$$S \subseteq \Omega$$

BASIC PROBABILITY

04

Expected value

CONTINUOUS RANDOM VARIABLE EXPECTED VALUE

$$\mathrm{E}(x) = \int_\Omega x dP(x) = \int_\Omega x p(x) dx$$

LINEARITY OF RANDOM VARIABLES WITH REGARDS TO E

$$\mathrm{E}(aX + bY) = a\mathrm{E}X + b\mathrm{E}Y$$

05

Variance

DEFINITION

$$\mathrm{Var}(X) = \mathrm{E}(X - \mathrm{E}X)^2$$

SUM OF I.I.D. RANDOM VARIABLE

$$\mathrm{Var}(aX + bY) = a^2\mathrm{Var}X + b^2\mathrm{Var}Y$$

BASIC PROBABILITY

05

Vector of random variables

SAMPLE MEAN

$$\overline{X} = \frac{X_1 + X_2 + \ldots + X_n}{n}$$

EXPECTED VALUE OF SAMPLE MEAN

$$\mathrm{E}\overline{X} = \frac{\mathrm{E}X_1 + \mathrm{E}X_2 + \ldots + \mathrm{E}X_n}{n} = \frac{n\mathrm{E}X}{n} = \mathrm{E}X$$

VARIANCE OF SAMPLE MEAN

$$\mathrm{Var}(\overline{X}) = \frac{\mathrm{Var}(X_1) + \mathrm{Var}(X_2) + \ldots + \mathrm{Var}(X_n)}{n^2} = \frac{n\mathrm{Var}(X)}{n^2} = \frac{\mathrm{Var}(X)}{n}$$

STANDARD DEVIATION

$$\sigma \propto \frac{1}{\sqrt{N}}$$

06

Strong law of large numbers

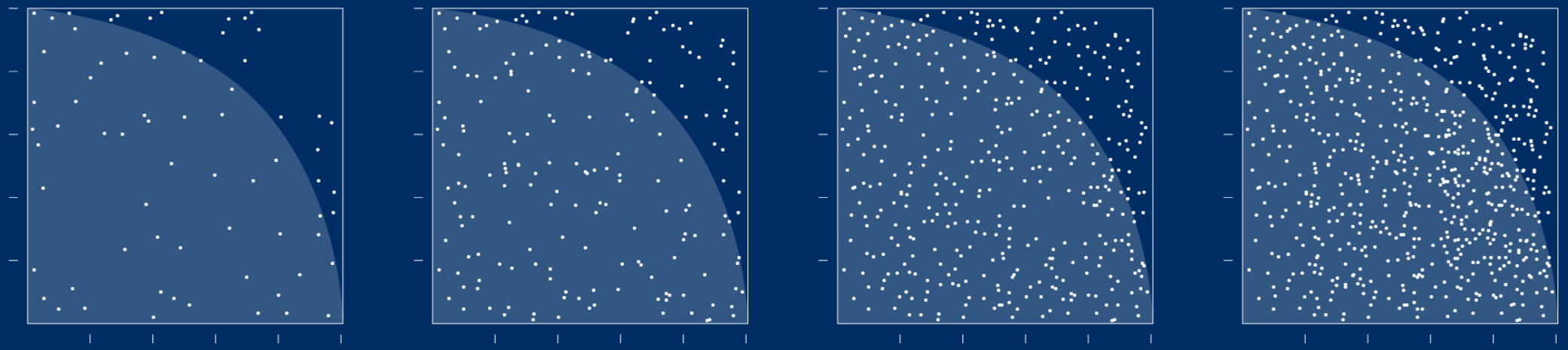
STRONG LAW OF LARGE NUMBERS

$$P(\mathrm{E}X = \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} X_k) = 1$$

MONTE CARLO METHODS

> Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.
>
> Can be used to solve any problem having a probabilistic interpretation.
>
> By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean of independent samples of the variable.

– WIKIPEDIA



APPROXIMATION OF π USING MC METHODS

MONTE CARLO INTEGRATION

01

Goal

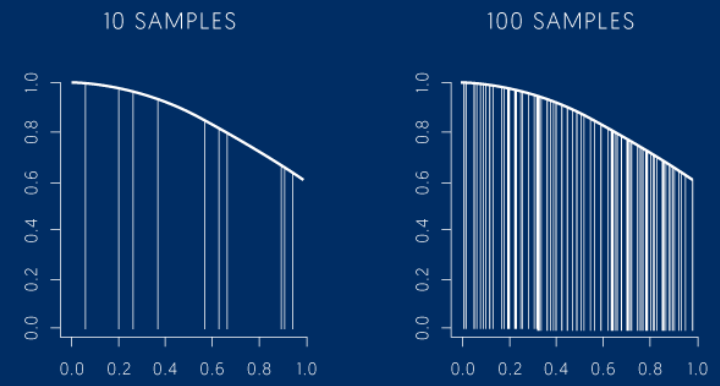
— Find the best possible approximation of the integral by sampling the integrated function at a finite set of points

02

Expected Value approximation

INTEGRAL

$$\int_S f(x)dx$$



AVERAGE WITH RESPECT TO SOME DENSITY P

$$\mathrm{E}(f(x)) = \int_\Omega f(x)dP(x) = \int_\Omega f(x)p(x)dx$$

STRONG LAW OF LARGE NUMBERS

$$P(\mathrm{E}X = \lim_{N\to\infty} \frac{1}{N}\sum_{k=1}^{N} X_k) = 1$$

AVERAGE ESTIMATION - UNIFORM SAMPLES

$$\mathrm{E}(f(x)) = \int_\Omega f(x)p(x)dx \approx \frac{1}{N}\sum_{k=1}^{N} f(x_k)$$

MONTE CARLO INTEGRATION

03

Mean of a function

04

Integral estimation using mean value

AN AVERAGE EXPRESSED AS AN EXPECTED VALUE WITH UNIFORM DENSITY FUNCTION

$$p(x) = \frac{1}{\mu(\Omega)}$$

$$\mathrm{E}(f(x)) = \int_\Omega \frac{1}{\mu(\Omega)} f(x) dx = \frac{1}{\mu(\Omega)} \int_\Omega f(x) dx = I$$

INTEGRAL VIA MEAN

$$\mu(\Omega) I = \mu(\Omega) \frac{1}{\mu(\Omega)} \int_\Omega f(x) dx = \int_\Omega f(x) dx$$

N UNIFORMLY DISTRIBUTED SAMPLES

$$x_k \sim U(\Omega)$$

INTEGRAL ESTIMATOR

$$\mu(\Omega) I = \mu(\Omega) \mathrm{E}(f(x)) \approx \mu(\Omega) \frac{1}{N} \sum_{k=1}^{N} f(x_k)$$

MONTE CARLO INTEGRATION

05

Integral expressed in terms of an arbitrary distribution

AVERAGE WITH RESPECT TO SOME DENSITY P

$$\mathbf{E}(f(x)) = \int_{\Omega} f(x)dP(x) = \int_{\Omega} f(x)p(x)dx$$

EXPECTED VALUE OF A SLIGHTLY MODIFIED FUNCTION

$$\mathbf{E}(\frac{f(x)}{p(x)}) = \int_{\Omega} \frac{f(x)}{p(x)}p(x)dx = \int_{\Omega} f(x)dx$$

A VECTOR OF N I.I.D. VARIABLES

$x_k \sim p$ where x_k independent identically distributred

INTEGRAL ESTIMATION - I.I.D. SAMPLES SAMPLED FROM THE DISTRIBUTION SPECIFIED BY P

$$\int_{\Omega} f(x)dx = \int_{\Omega} \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{N}\sum_{k=1}^{N} \frac{f(x_k)}{p(x_k)}$$

MONTE CARLO INTEGRATION

06

Problem of diminishing return

07

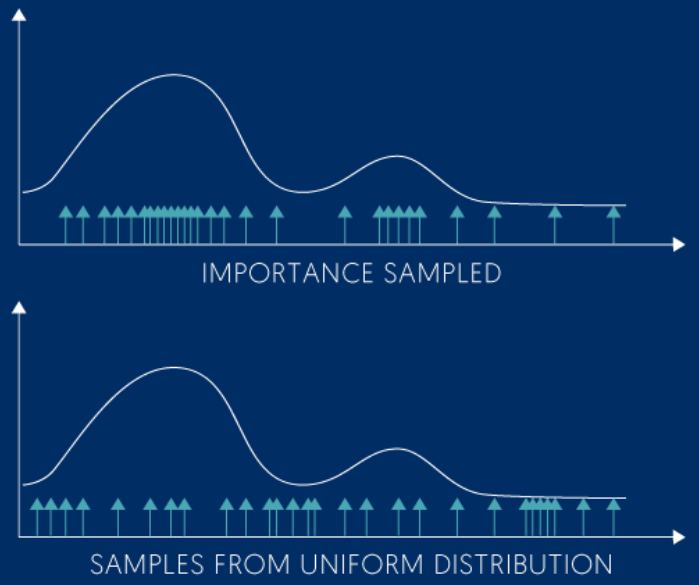
Variance reduction techniques

— Choosing density wisely we can greatly reduce initial variance (**Importance Sampling**)

STANDARD DEVIATION

$$\sigma \propto \frac{1}{\sqrt{N}}$$

> To halve the error the number of samples has to be quadrupled.



MONTE CARLO INTEGRATION

06

Problem of diminishing return

STANDARD DEVIATION

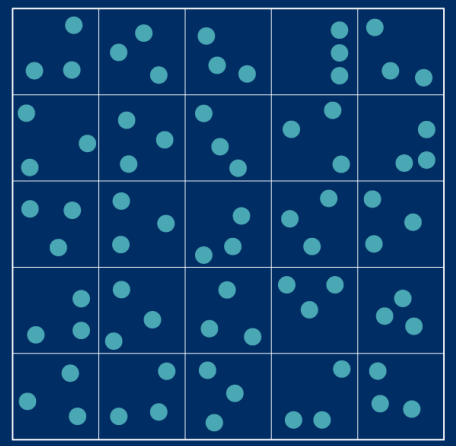
$$\sigma \propto \frac{1}{\sqrt{N}}$$

> To halve the error the number of samples has to be quadrupled.

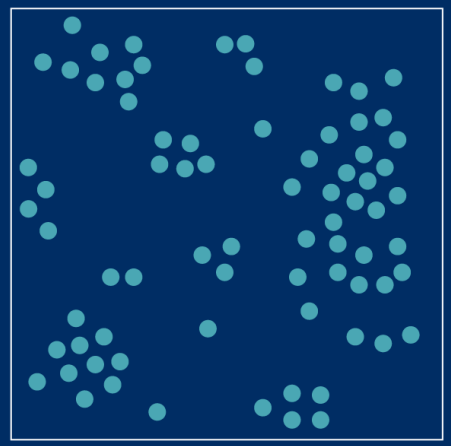
07

Variance reduction techniques

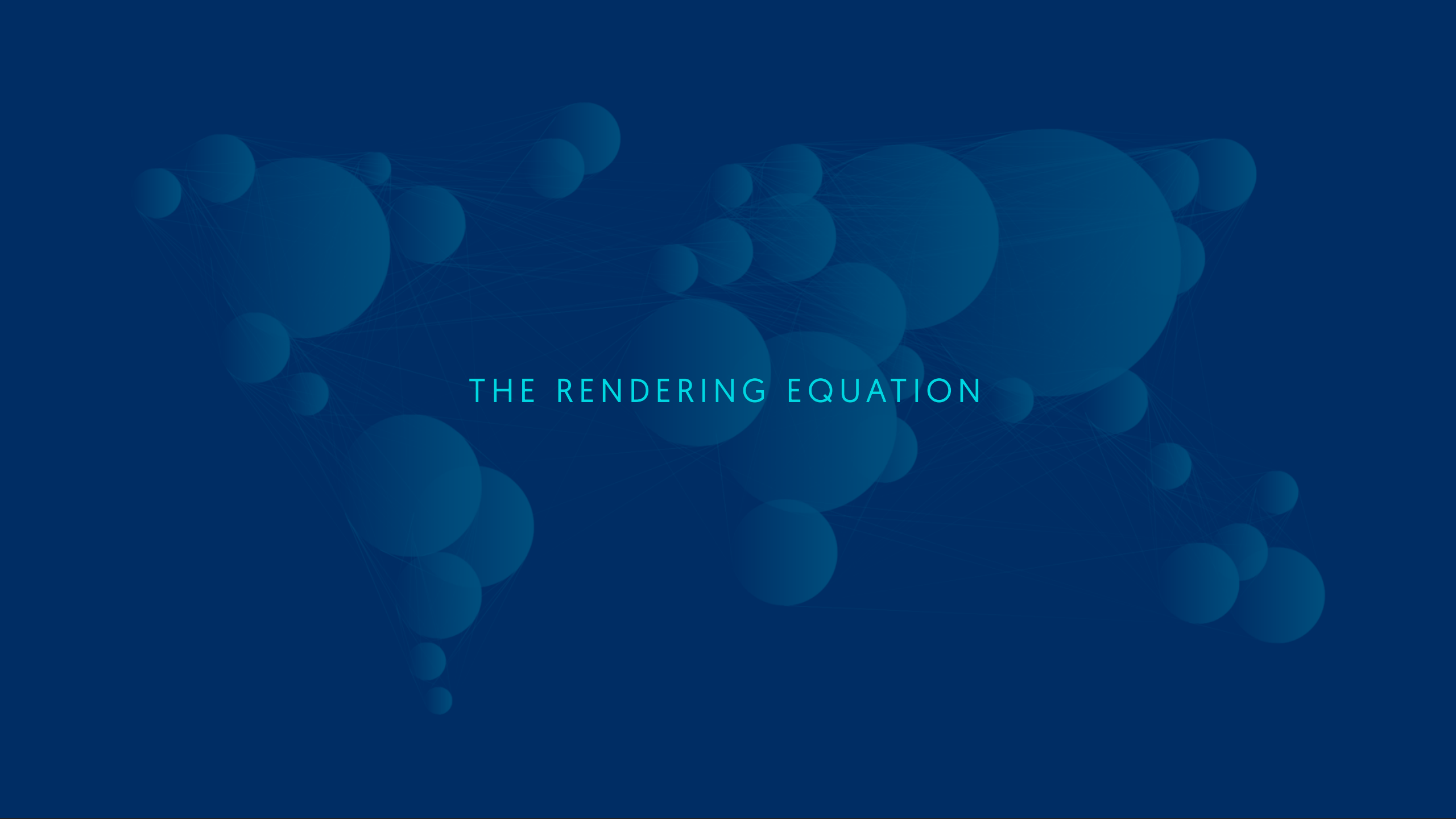
- Domain partitioning (**Stratified Sampling**)



STRATIFIED



UNIFORM

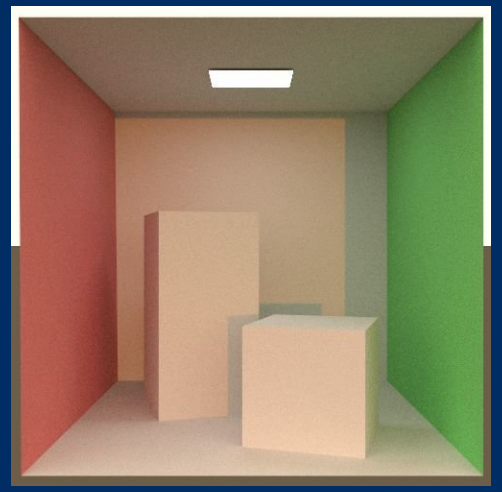

THE RENDERING EQUATION

RENDERING EQUATION

> In computer graphics, the rendering equation is an integral equation in which the equilibrium radiance leaving a point is given as the sum of emitted plus reflected radiance under a geometric optics approximation.
>
> – WIKIPEDIA

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + \int_\Omega \rho(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') \cos\theta d\vec{\omega}'$$

$$\cos\theta = (\vec{\omega}' \cdot \vec{n})$$



RADIOMETRIC QUANTITIES

01

Energy

- Energy of a collection of photons

ENERGY

$$Q$$

02

Flux

- Time rate of flow of radiant energy
- This quantity expresses how much total energy flows from/to/through a surface per unit time

FLUX

$$\Phi = \frac{dQ}{dt}$$

03

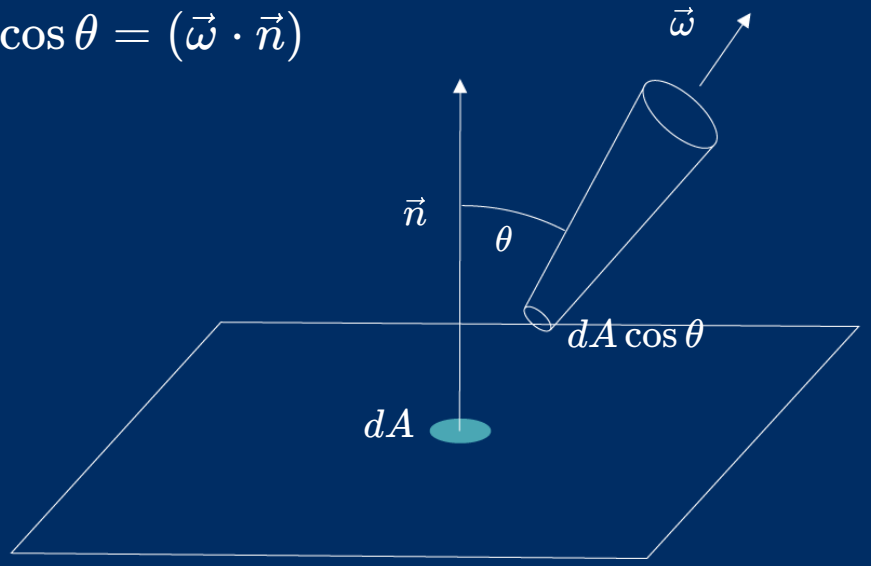
Radiance

- Radiant flux per unit solid angle per unit projected area
- Intuitively, radiance expresses how much power (flux) arrives at (or leaves from) a certain point on a surface, per unit solid angle, and per unit projected area

RADIANCE

$$L(x, \vec{\omega}) = \frac{d^2\Phi}{\cos\theta dA d\vec{\omega}}$$

$$\cos\theta = (\vec{\omega} \cdot \vec{n})$$

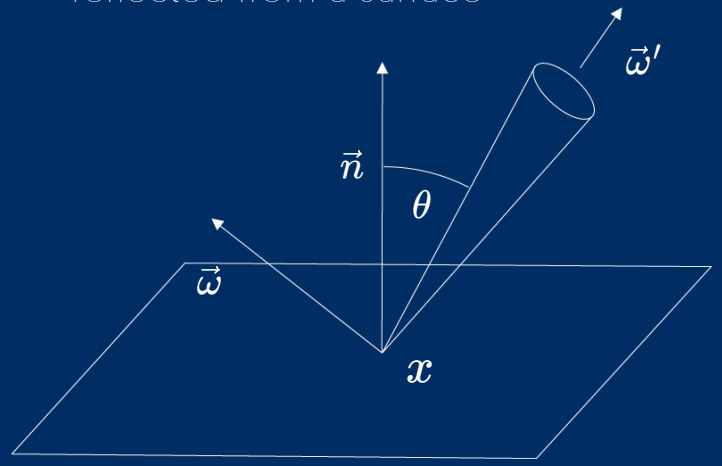


RADIANCE PROPAGATION

01

Bidirectional Reflectance Distribution Function (BRDF)

— A function that defines how light is reflected from a surface



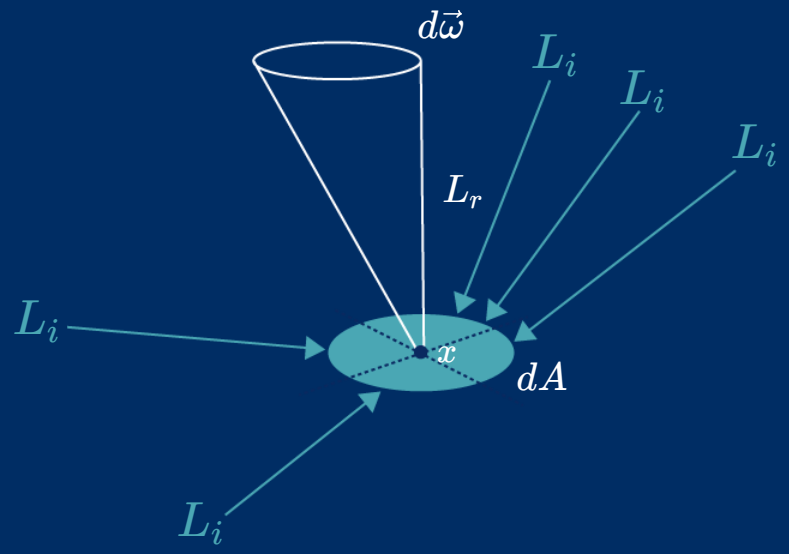
BRDF

$$\rho(x, \vec{\omega}', \vec{\omega}) = \frac{dL_r(x, \vec{\omega})}{L_i(x, \vec{\omega}') \cos\theta d\vec{\omega}'}$$

$$\cos\theta = (\vec{\omega}' \cdot \vec{n})$$

02

Reflected radiance



DIFFERENTIAL REPRESENTATION

$$dL_r(x, \vec{\omega}) = \rho(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') \cos\theta d\vec{\omega}'$$

INTEGRAL FORM

$$L_r(x, \vec{\omega}) = \int_\Omega \rho(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') \cos\theta d\vec{\omega}'$$

RENDERING EQUATION

01

Outgoing radiance

02

Rendering equation

03

Operator formulation

OUTGOING RADIANCE

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + L_r(x, \vec{\omega})$$

RENDERING EQUATION

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + \int_\Omega \rho(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}') \cos\theta d\vec{\omega}'$$

$$\cos\theta = (\vec{\omega}' \cdot \vec{n})$$

INTEGRAL FORM IN OPERATOR NOTATION

$$T = L_e + TL$$

OPERATOR DEFINED AS

$$< Tg > (x, \vec{\omega}) = \int_\Omega \rho(x, \vec{\omega}', \vec{\omega}) g(x, \vec{\omega}') \cos\theta d\vec{\omega}'$$

$$\cos\theta = (\vec{\omega}' \cdot \vec{n})$$

RENDERING EQUATION

04

Neumann Series

- Renderers - black box engines
  - The rendering equation expressed via Neumann series expansion suggests that the final solution can be composed of independently computed results.
  - These partial results are computed by renderers, which can be treated as black box computation engines

NEUMANN SERIES EXPANSION

$$T = L_e + TL_e + T^2L_e + T^3L_e + \dots$$

COMPACT NOTATION

$$T = \sum_{k=0}^{\infty} T^k L_e$$

RENDERING PROCESS

RENDERING

> Rendering or image synthesis is the automatic process of generating a photorealistic or non-photorealistic image from a 2D or 3D model (or models in what collectively could be called a *scene* file) by means of computer programs. Also, the results of displaying such a model can be called a render. A scene file contains objects in a strictly defined language or data structure; it would contain geometry, viewpoint, texture, lighting, and shading information as a description of the virtual scene. The data contained in the scene file is then passed to a rendering program to be processed and output to a digital image or raster graphics image file.
>
> – WIKIPEDIA

RENDERER

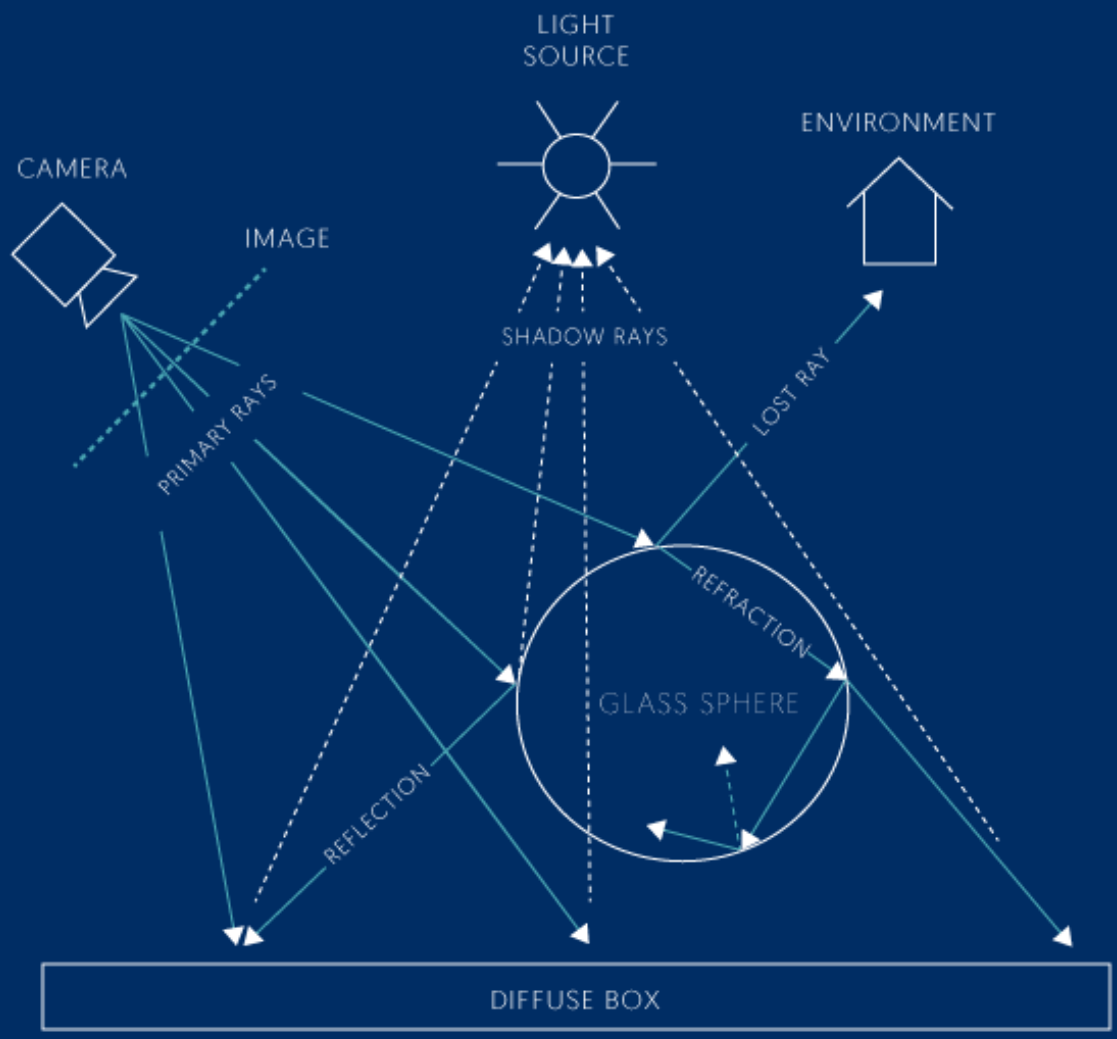
> (in this context) a computer program used to generate photorealistic images from a scene file (consisting of 3D models and description of materials and lights)

PATH TRACING

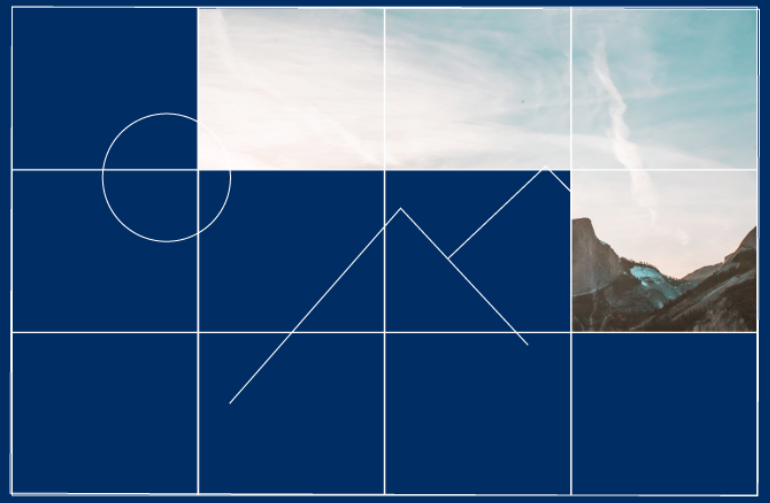
01

Path formulation

- Markov Chain approach
- Independent evaluation of each path



EXAMPLES OF PATH GROUPING



- Each tile is rendered independently, the final image is composed of tiles rendered with full quality

- Multiple, independent full resolution images are rendered. Each one uses only a fraction of samples per pixel as compared to the final result. The final image is obtained by means of averaging these intermediate results

MONTE CARLO METHODS

Verification challenges

VERIFICATION CHALLENGES

01

Game-theoretical approach

- Each rational participant wants to maximize his/her benefit

02

Verification process outline

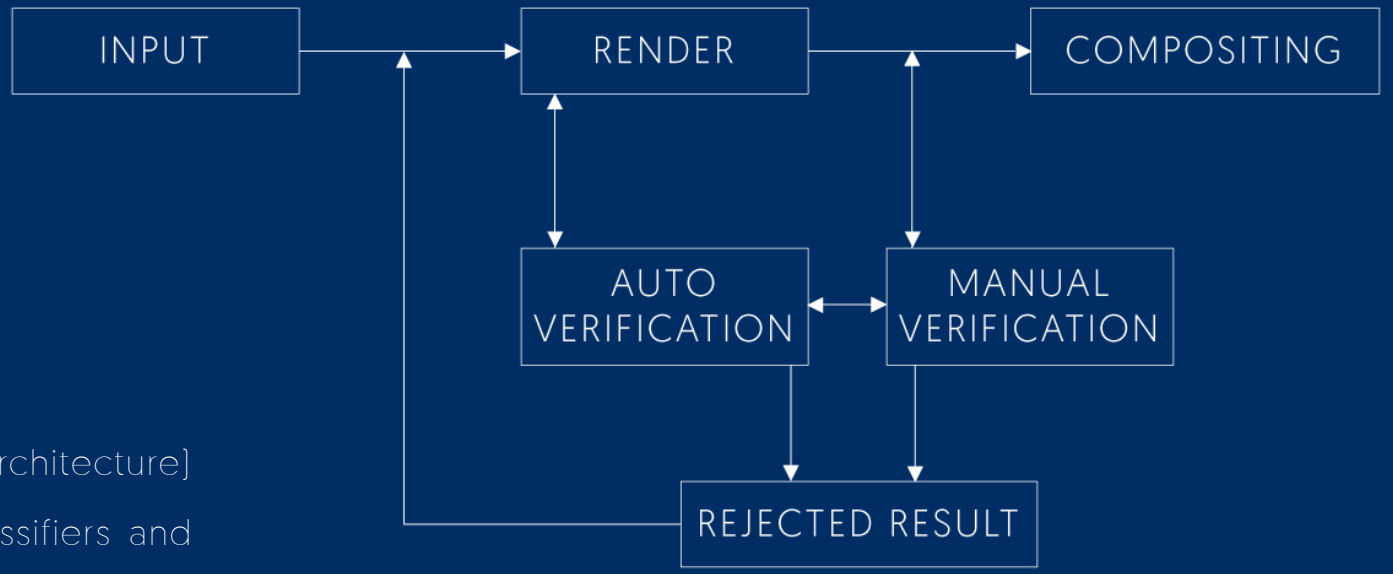
- 1st step - automatic verification

  Classifier can be easily replaced (it is a plugin architecture)

  The main classifier can consist of multiple classifiers and image comparison algorithms

- 2nd step - manual verification and acceptance/rejection

HYBRID APPROACH



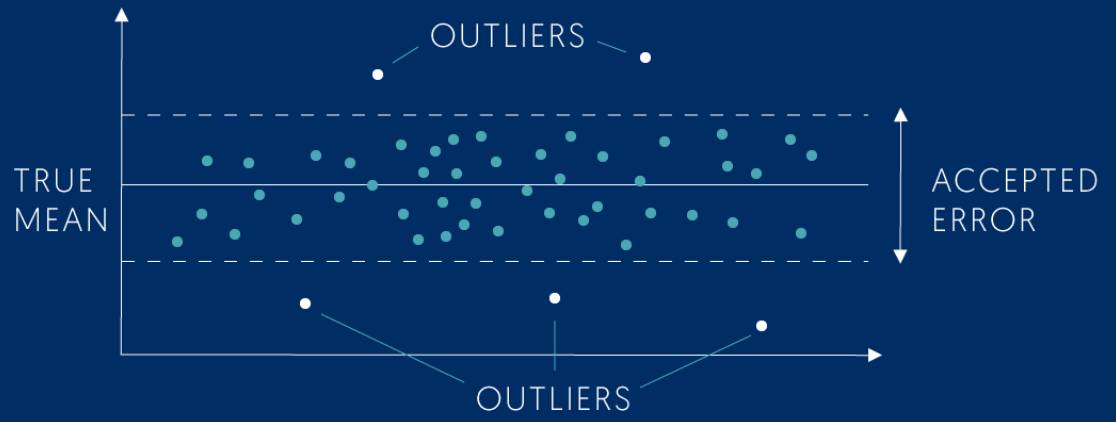
VERIFICATION CHALLENGES

03

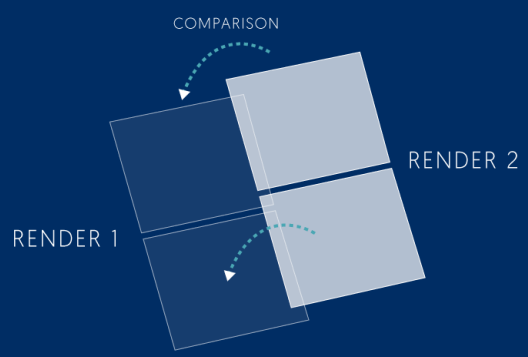
Automatic verification

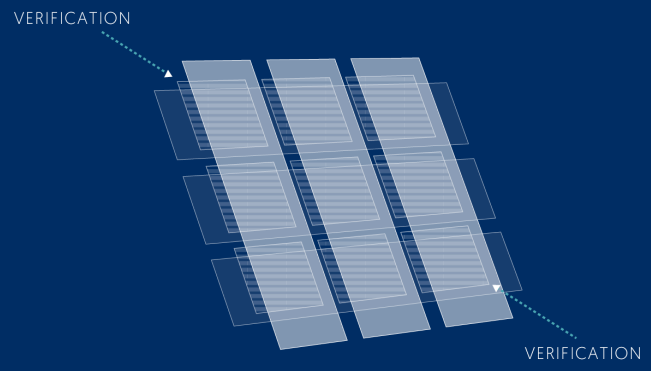
- Outlier removal method
- Verification based on redundant computations (probabilistic)

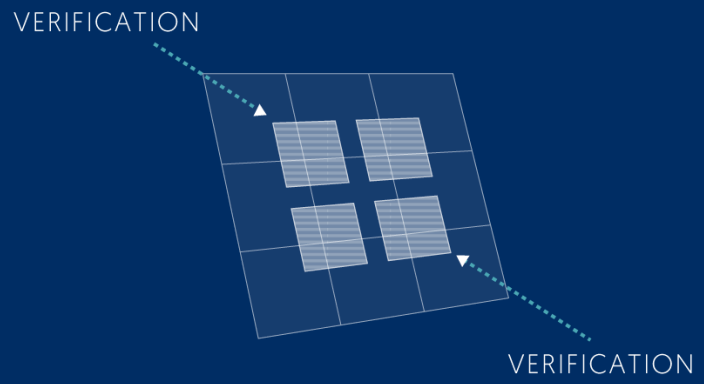
OUTLIER REMOVAL METHOD



VERIFICATION BASED ON REDUNDANT COMPUTATIONS







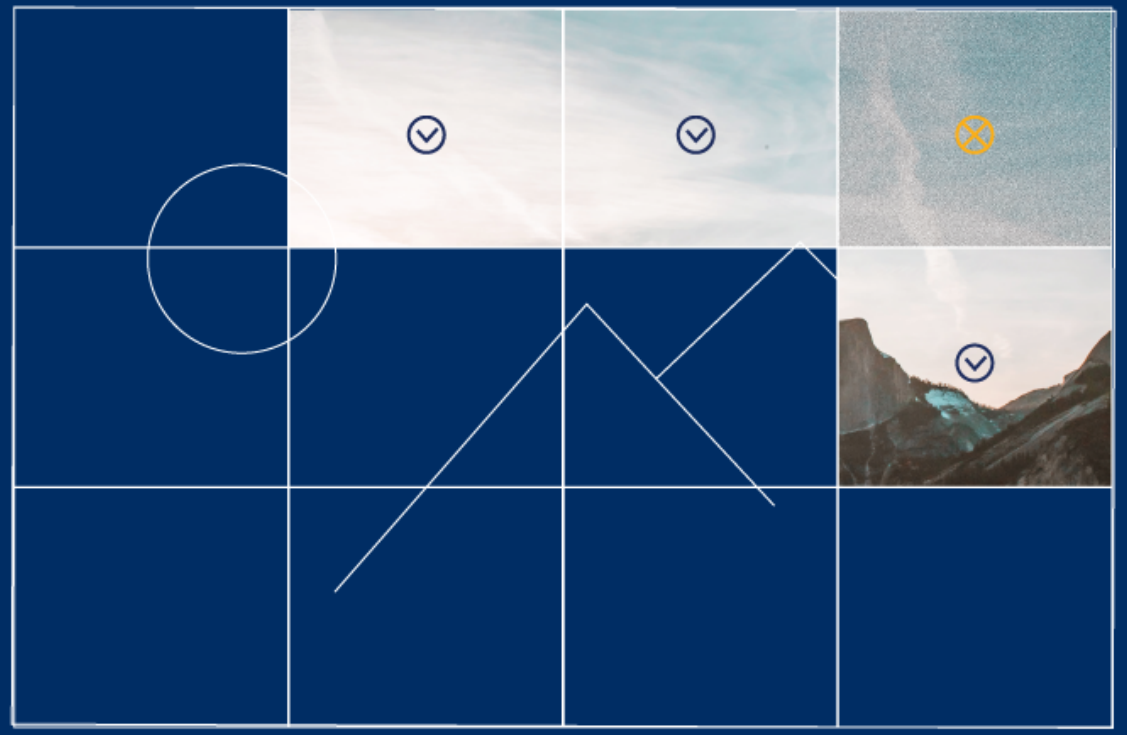
VERIFICATION CHALLENGES

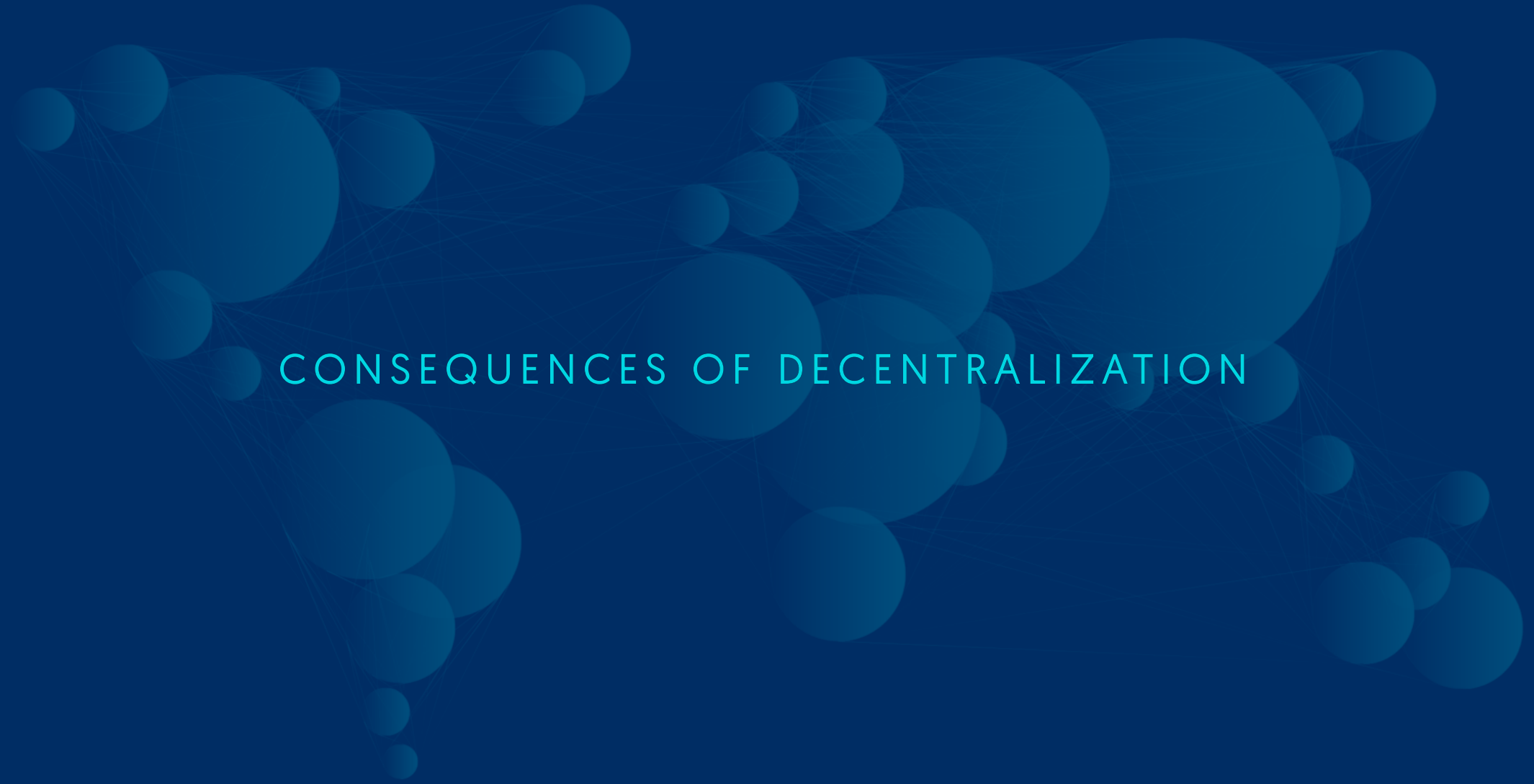
04

Manual verification

— The final result may be accepted or rejected manually

MANUAL VERIFICATION




CONSEQUENCES OF DECENTRALIZATION

CONSEQUENCES OF DECENTRALIZATION

DEFAULT APPROACH

01

Requestor prepares scenes and sends them together with the resources to a render farm

02

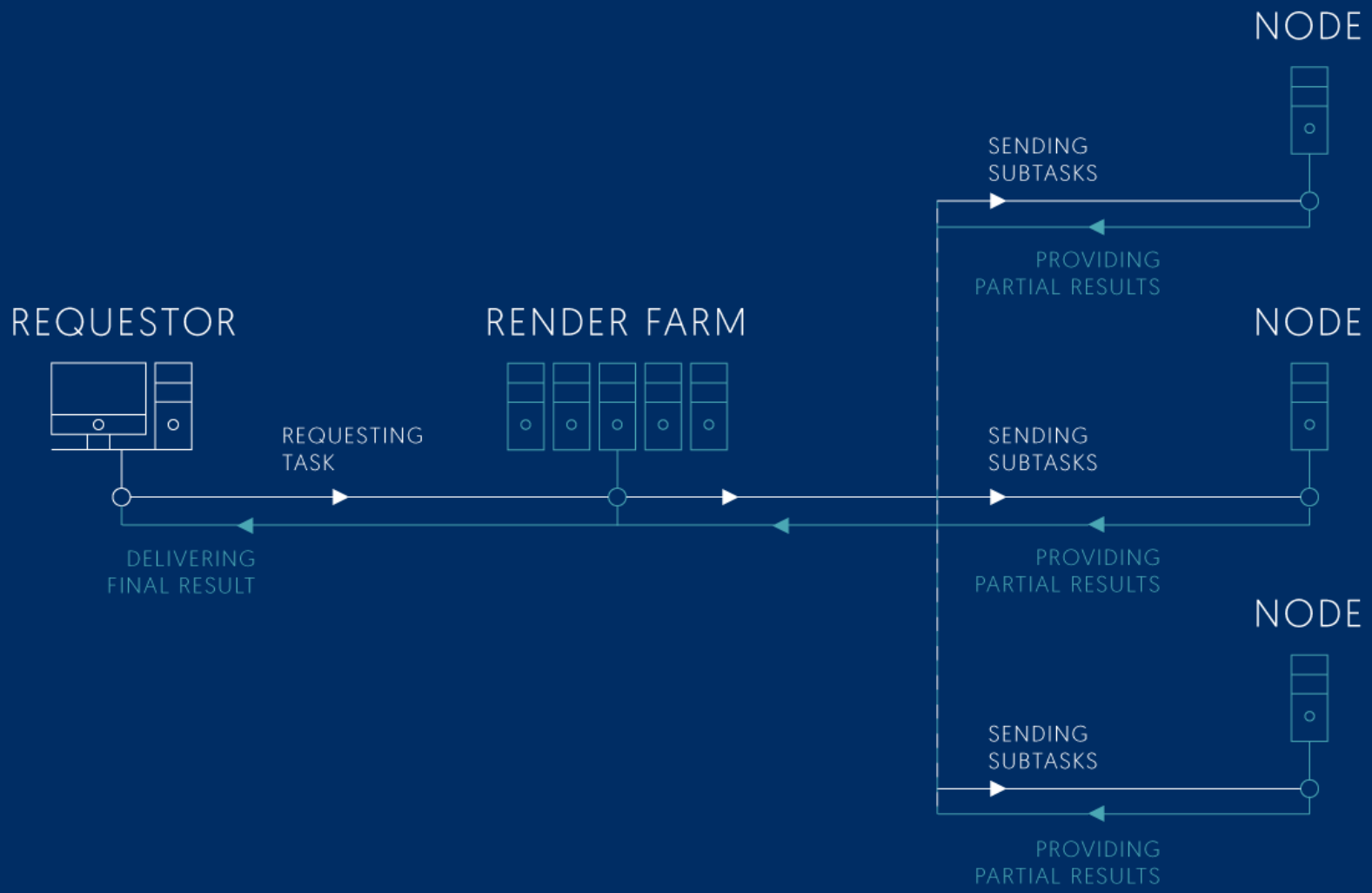
Render farm splits the task and distributes computations between nodes

03

Nodes compute partial results which are composed into final images/animations

04

The results are sent back to the requestor



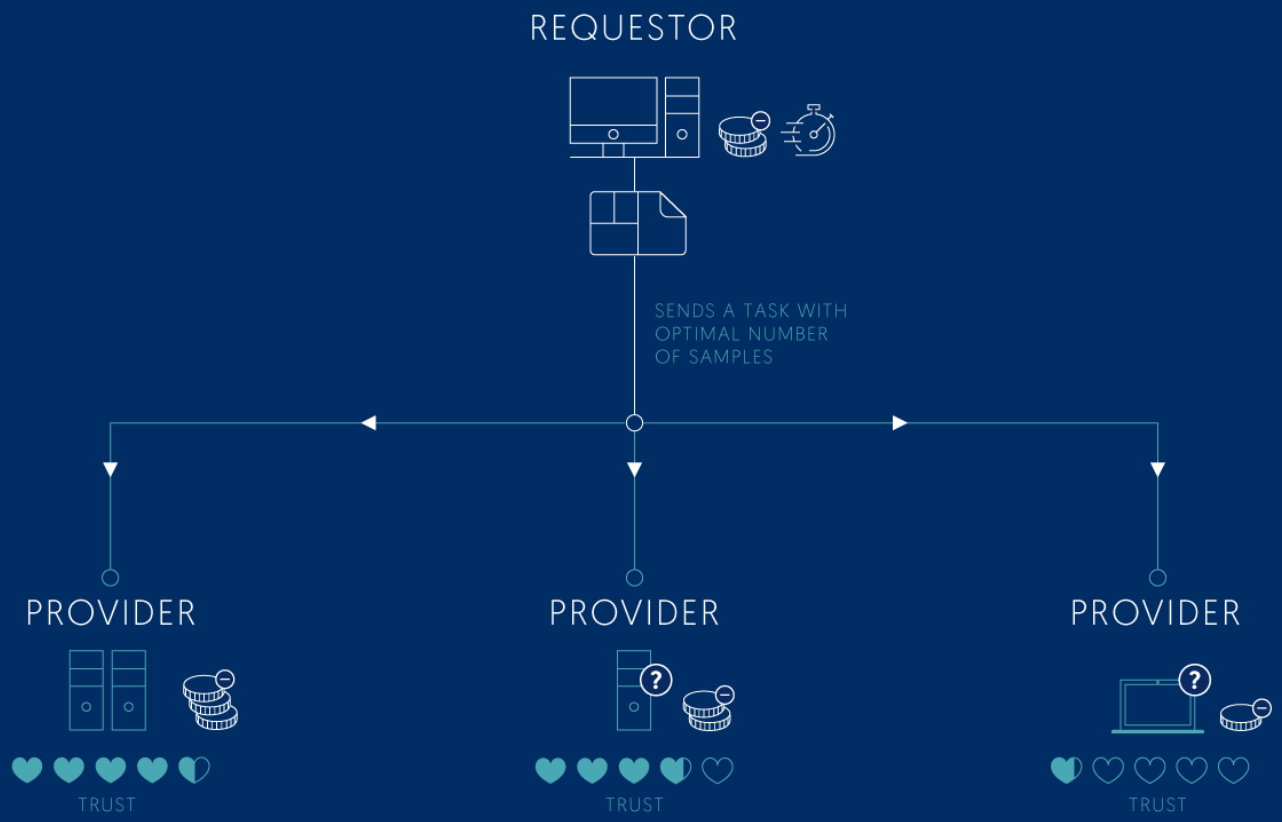
CONSEQUENCES OF DECENTRALIZATION

DECENTRALIZED SETTING

01

Requestor wants a valid result for the lowest price possible in a specified timeframe:

- Must carefully set the number of samples for a valid result to avoid unnecessary computation
- Won't pay for invalid results
- Requires fair protocol in case of invalid results



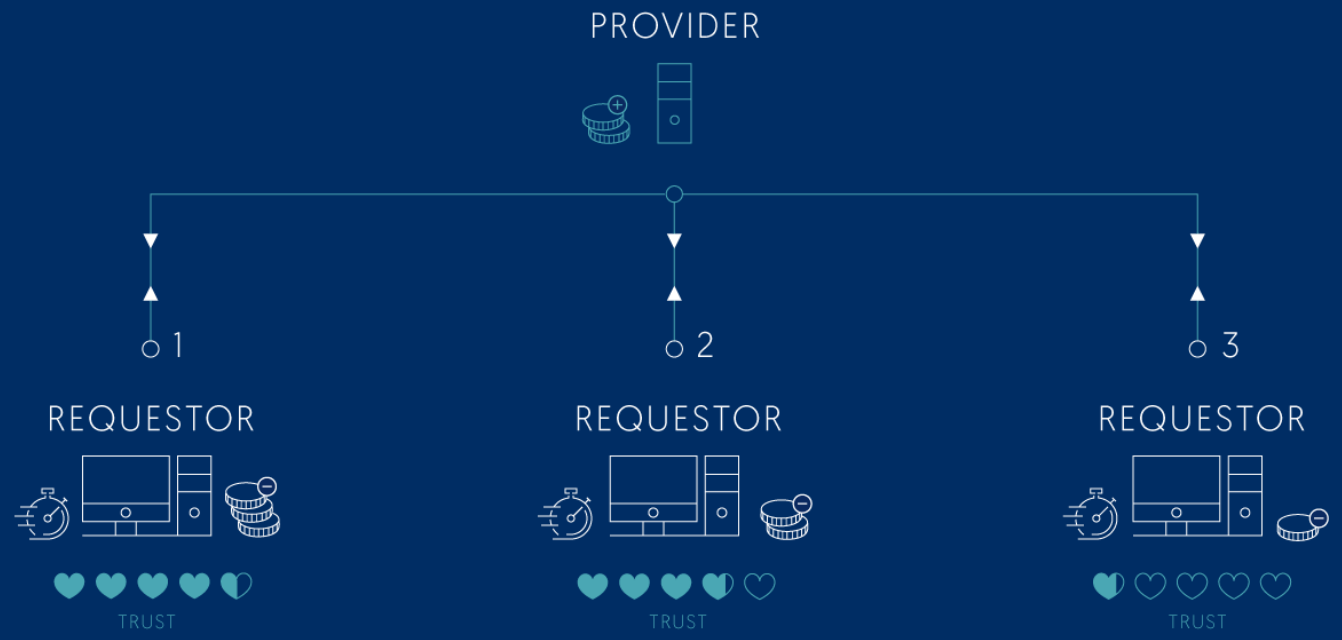
CONSEQUENCES OF DECENTRALIZATION

DECENTRALIZED SETTING

02

Provider wants to earn as much as possible for the least possible resources provided:

- Can try to generate the image of lesser quality and use fewer samples than the requestor requested
- May not be paid for valid results and needs to have a way of recovering the payment
- Protocol must be fair



VERIFICATION IN THE CASE OF MONTE CARLO RENDERING

VERIFICATION - MONTE CARLO RENDERING EXAMPLE

01

Blender (Cycles renderer) use-case

- Bit-by-bit verification is not viable
- Each tile has to be either marked as valid or invalid by automatic verification algorithm

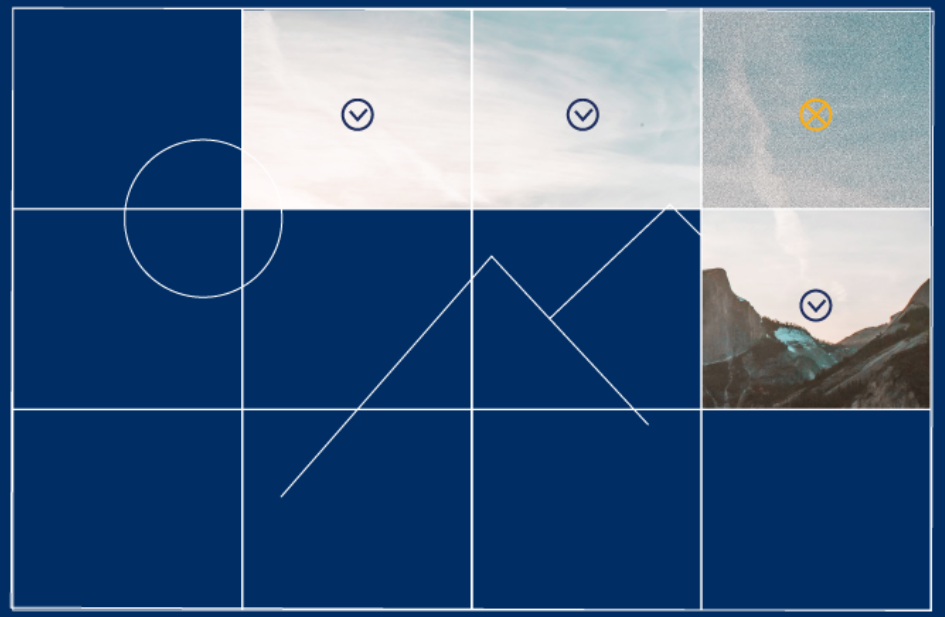
VERIFICATION - MONTE CARLO RENDERING EXAMPLE

02

Example of classifier implementation

- Local verification
  - Sample rectangles are rendered with the same parameters as the main tile
  - Each sample must match the corresponding rectangle in the tile
  - Total area of sample rectangles is only a small fraction of the tile area

AUTOMATIC VERIFICATION

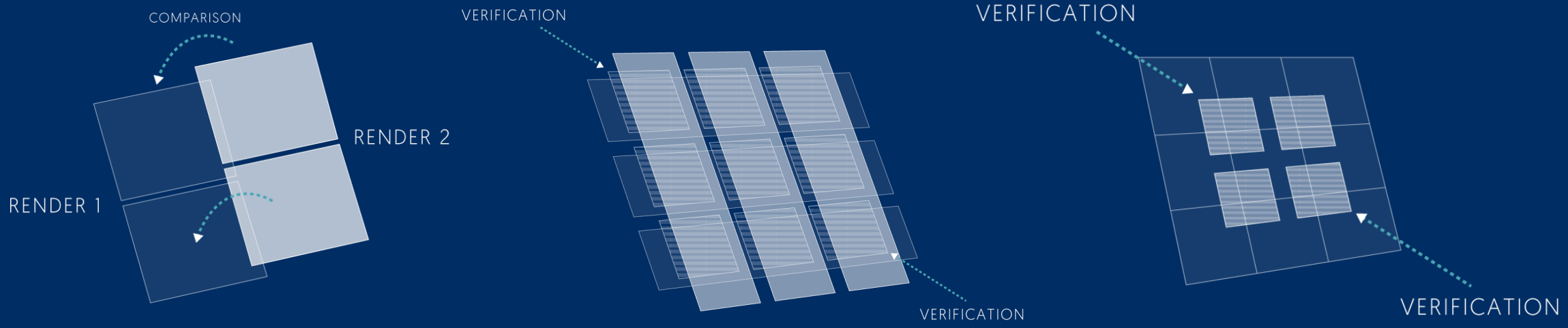


VERIFICATION - MONTE CARLO RENDERING EXAMPLE

03

Example of classifier implementation

- Redundancy
  - Rendering each tile multiple times and comparing
  - Rendering redundantly but not necessarily duplicating all pixels
  - Multiple schemes are possible, which makes it harder to cheat



VERIFICATION - MONTE CARLO RENDERING EXAMPLE

SUMMARY

01

It is probabilistic

02

Classifier only assists

03

It is adjustable

- Tunable redundancy
- Tunable verification thresholds
- Tunable trust requirements (e.g. when SGX nodes are present)

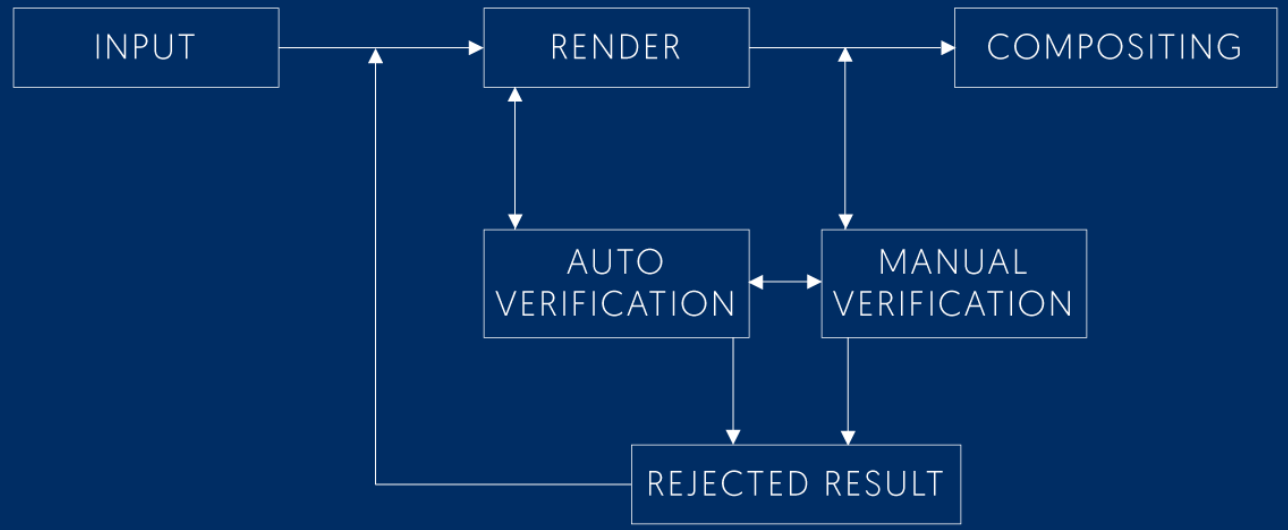


IMAGE COMPARISON AND VERIFICATION

IMAGE COMPARISON AND VERIFICATION

01

Formal problem statement

VERIFICATION PREREQUISITES

$b \in B$ input bitmap

$R \in \Re$ renderer

$q \in P$ unknown parameters used to render b

$p \in P$ parameters specified by the requestor

— For the same input parameters and renderer run in exactly the same conditions we have

$$p = q$$

$$R(q) = R(p) = b$$

02

Estimation

APPROACH 1 - ESTIMATE THE PROBABILITY

$$P(q = p)$$

provided that we know

p, b, R

APPROACH 2 - ESTIMATE THE PROBABILITY (PREFERRED SOLUTION)

$$P(R(p) = b)$$

provided that we know

p, b, R

but we do not want to recompute the whole

$R(p)$

IMAGE COMPARISON AND VERIFICATION

01

Similarity of images

- a decent algorithm is required to help the classifier decide if two bitmaps are similar enough to be treated as the result of the same computation

02

Example approach

- Two transforms
  - Wavelet transform
  - Edge enhancing transform
- Two measures
  - SSIM (structural similarity)
  - MSE (mean squared error)

TRANSFORM

$$T_j : B \to B$$

SIMILARITY MEASURE

$$M_i : B \times B \to \mathbb{R}$$

IMAGE COMPARISON AND VERIFICATION

03

Importance of the transforms

- Picture w/ no transform vs Picture w/ edge enhancing transform (one SPP difference in images)

COMPARISON OF TWO BITMAPS

$$m_{ji} = M_i(T_j(b_1), T_j(b_2))$$

PICTURE W/ NO TRANSFORM

IMAGE COMPARISON AND VERIFICATION

03

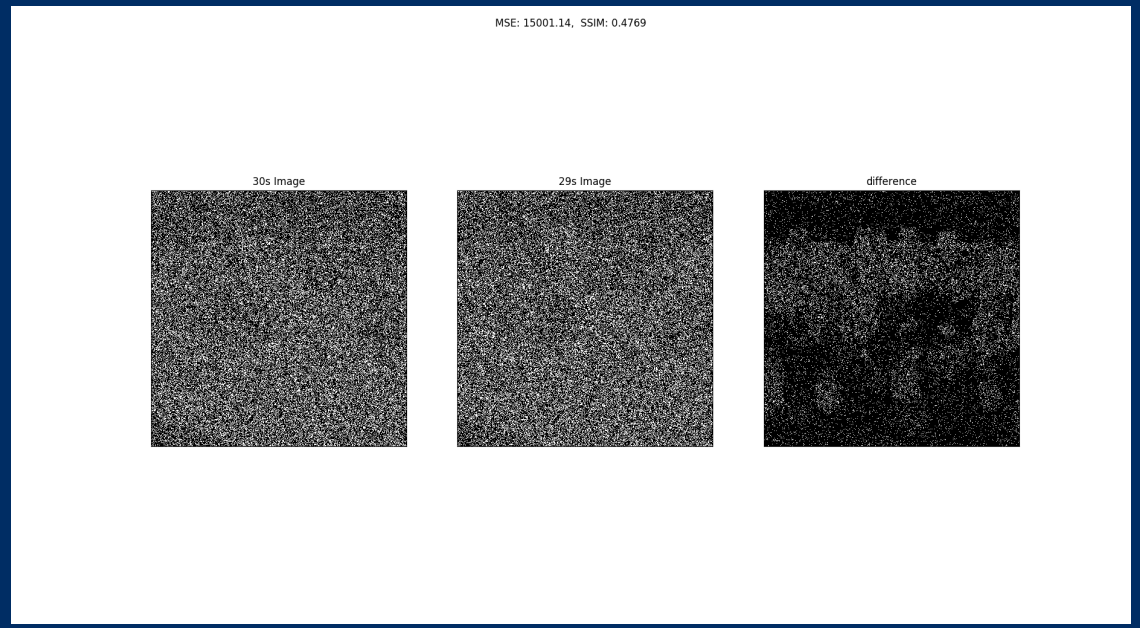
Importance of the transforms

- Picture w/ no transform vs Picture w/ edge enhancing transform (one SPP difference in images)

COMPARISON OF TWO BITMAPS

$$m_{ji} = M_i(T_j(b_1), T_j(b_2))$$

PICTURE W/ EDGE ENHANCING TRANSFORM

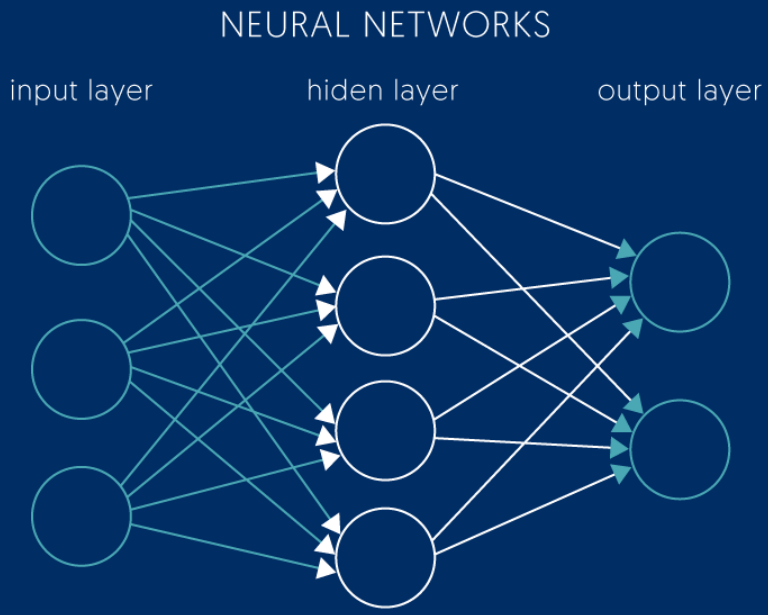


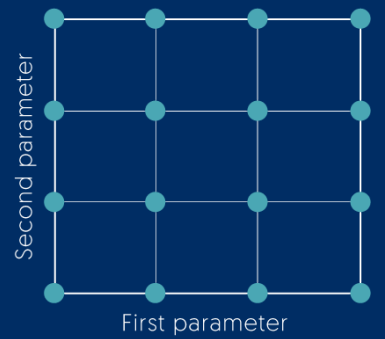
OTHER USE-CASES

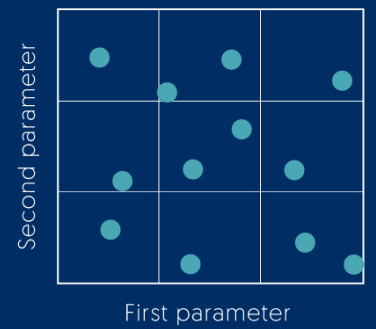
01

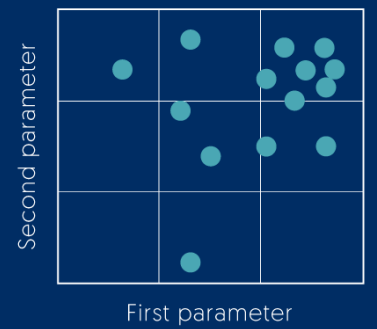
Machine Learning

- Golem can be used to find the best set of hyperparameters for a given model
- Hyperparameters describe network structure, eg. number of layers, number of nodes in each layer. They are set before training
- Each provider is doing a full training of a neural network but for different values of hyperparameters
- Requestors can use different algorithms for searching parameter space, eg. grid search, random search, bayesian optimization search
- Each dot in the picture represents one set of parameters, ie. one network trained by Provider







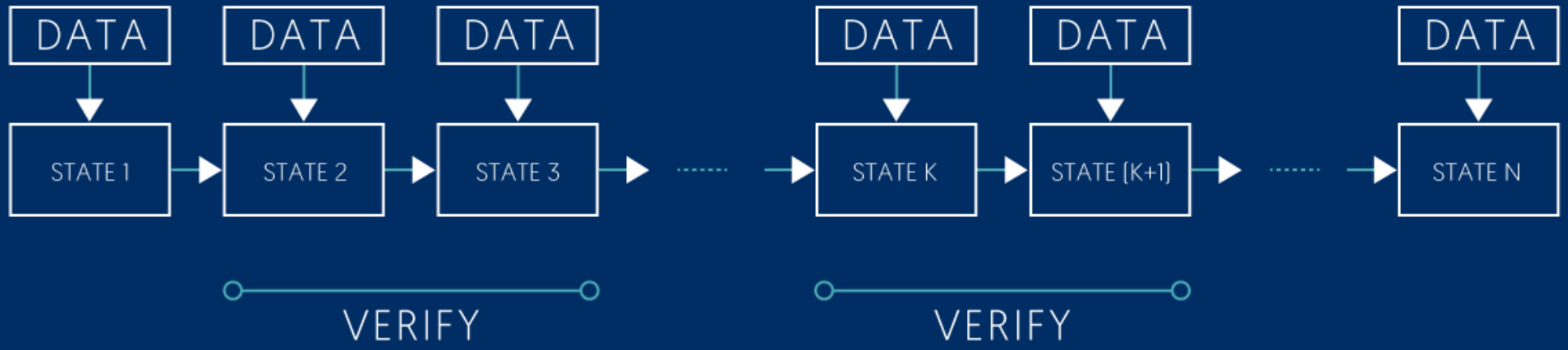


OTHER USE-CASES

02

Machine Learning - Verification

- Neural network training is a sequential algorithm
- Feeding sets of batches to the network is one iteration step. After every step, the provider computes the hash of the network state and sends it to requestor
- Requestor asks provider for randomly chosen states and verifies that transition to the next state was done correctly
- Each rectangle represents one network state. The Requestor checks the transition between randomly chosen states



OTHER USE-CASES

01

Proof of Work

NONCES FOR POW

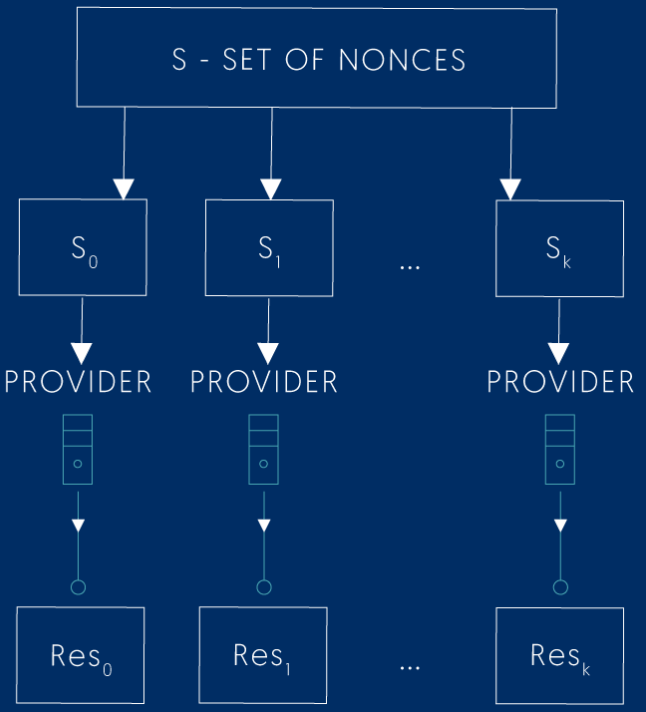
$$S = \{0, 1, \ldots, 2^{n-1}\}$$

$$\sigma = \begin{pmatrix} 0, & 1, & 2, & \ldots, 2^{n-1} \\ \sigma(0), & \sigma(1), & \sigma(2), & \ldots, \sigma(2^{n-1}) \end{pmatrix}$$

$$\underbrace{\qquad}_{S_1} \quad \cdots \quad \underbrace{\qquad}_{S_K}$$

$$\bigcup_{i=1}^{K} S_i = S$$

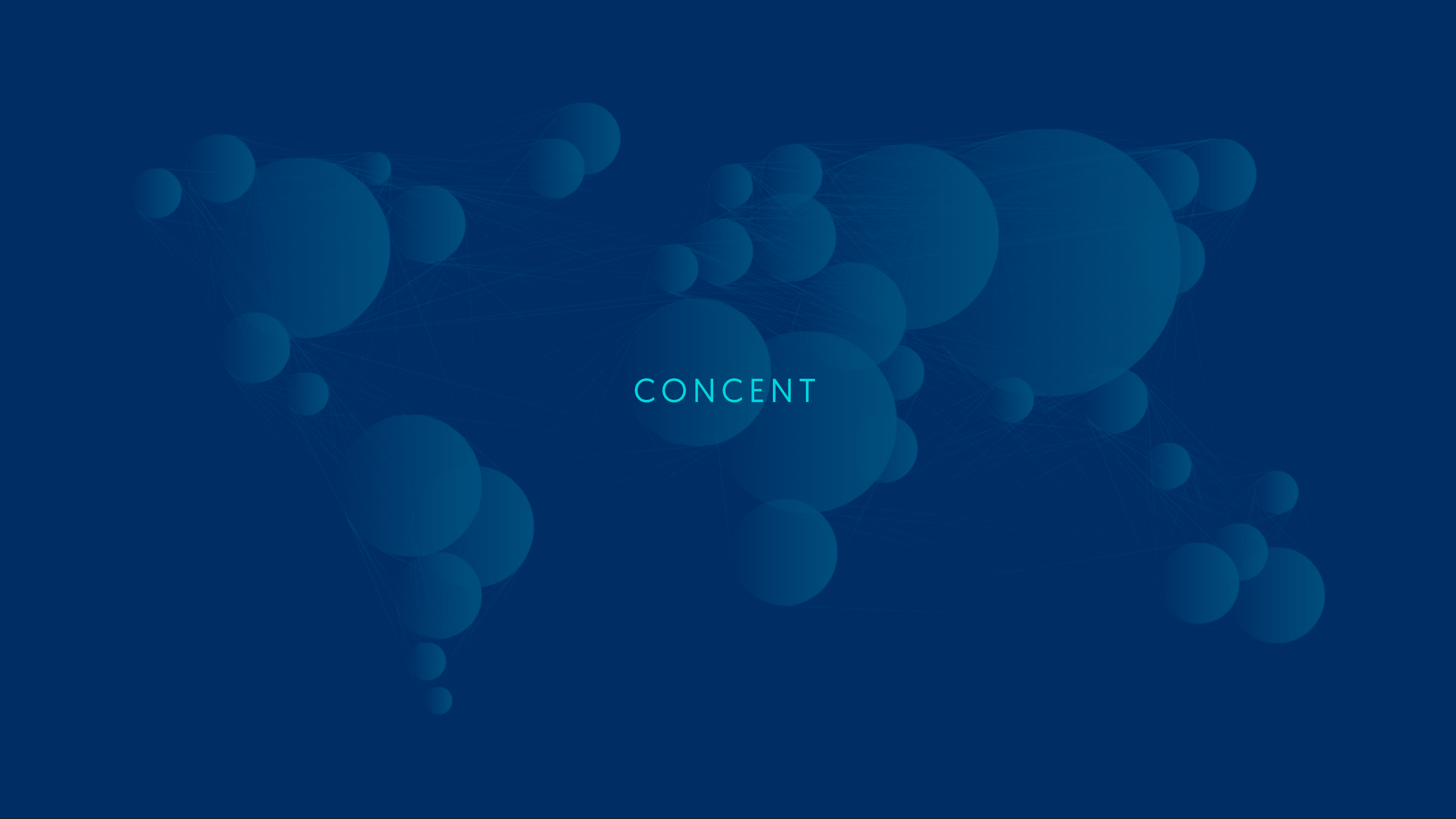
$$S_i \cap S_j = \emptyset \text{ for } i \neq j$$



POW TARGETS

$$\text{Res}_i = \{\{k \in S_i | H(k \oplus \omega) \leq t\}, \{n \in S_i | H(n \oplus \omega) \leq T\}\}$$

- ω PoW input string
- H PoW hash function
- T PoW target
- t PoW intermediate target (used during verification), $T < t$

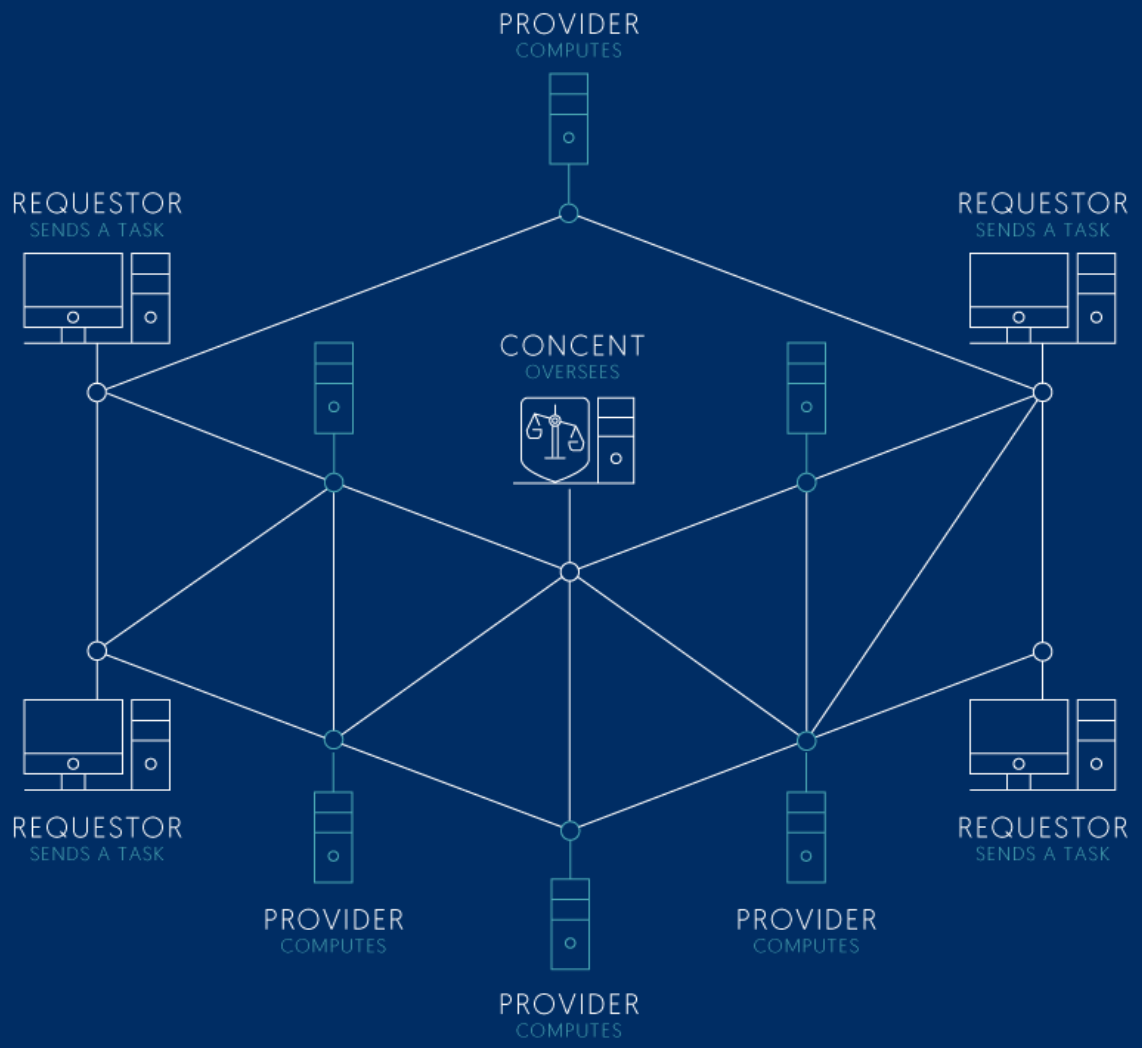

CONCENT

WHAT IS CONCENT?

- Providers should be paid for calculations they do
- Requestors must get proper results (honest calculations)
- If something goes wrong, an optional third party may be utilized to solve the dispute

WHAT IS CONCENT?

- It is a web service (which ultimately may be distributed)
- It is a special type of node in the Golem Network
- Its purpose is to improve the fairness of transactions
- It is passive
- It is optional

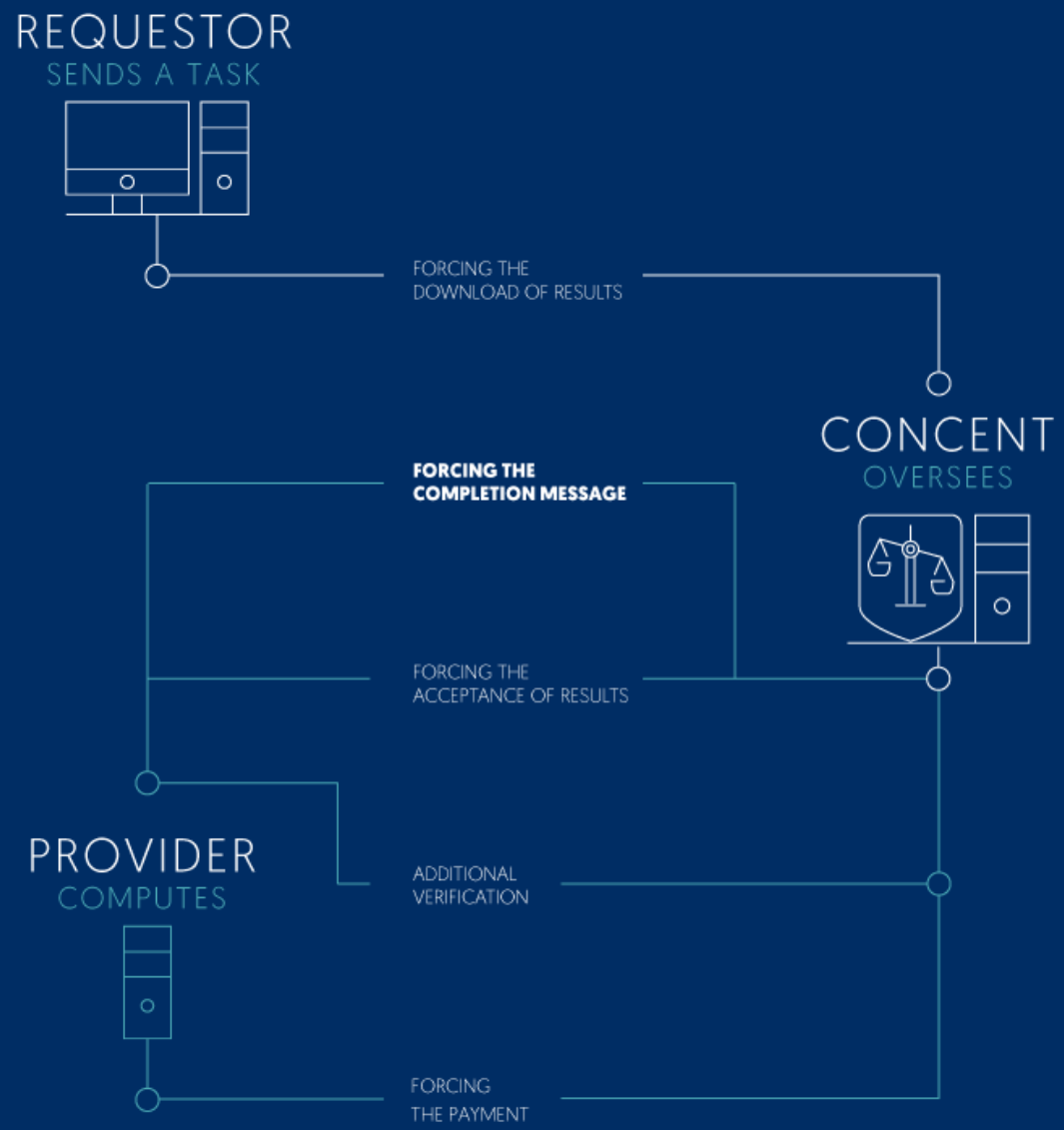


WHAT CONCENT DOES

01

Enforcing communication:

- After a Requestor gives a task to a Provider and the Provider starts to compute, the communication must follow the protocol. If one party goes offline or rejects accepting proper messages, Concent can enforce the right communication, or ensure that the malicious party takes responsibility for the broken communication

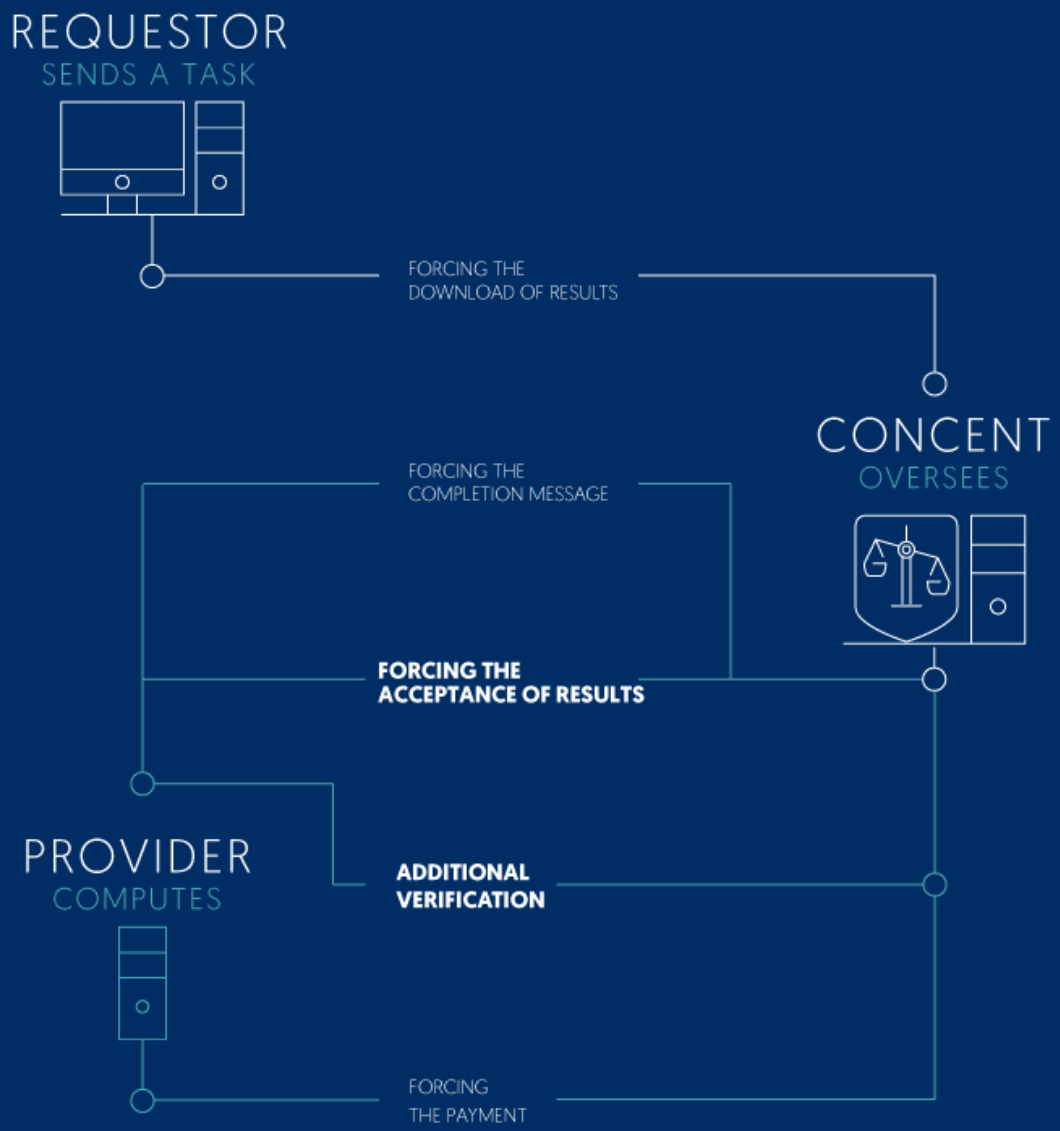


WHAT CONCENT DOES

02

Additional verification:

— Verification is performed by Requestors. If a Requestor rejects the correct result, the Provider calls Concent which performs additional verification. It is similar to the one performed by the Requestor but stronger. If the result turns out to be correct then Concent forces the Requestor to pay

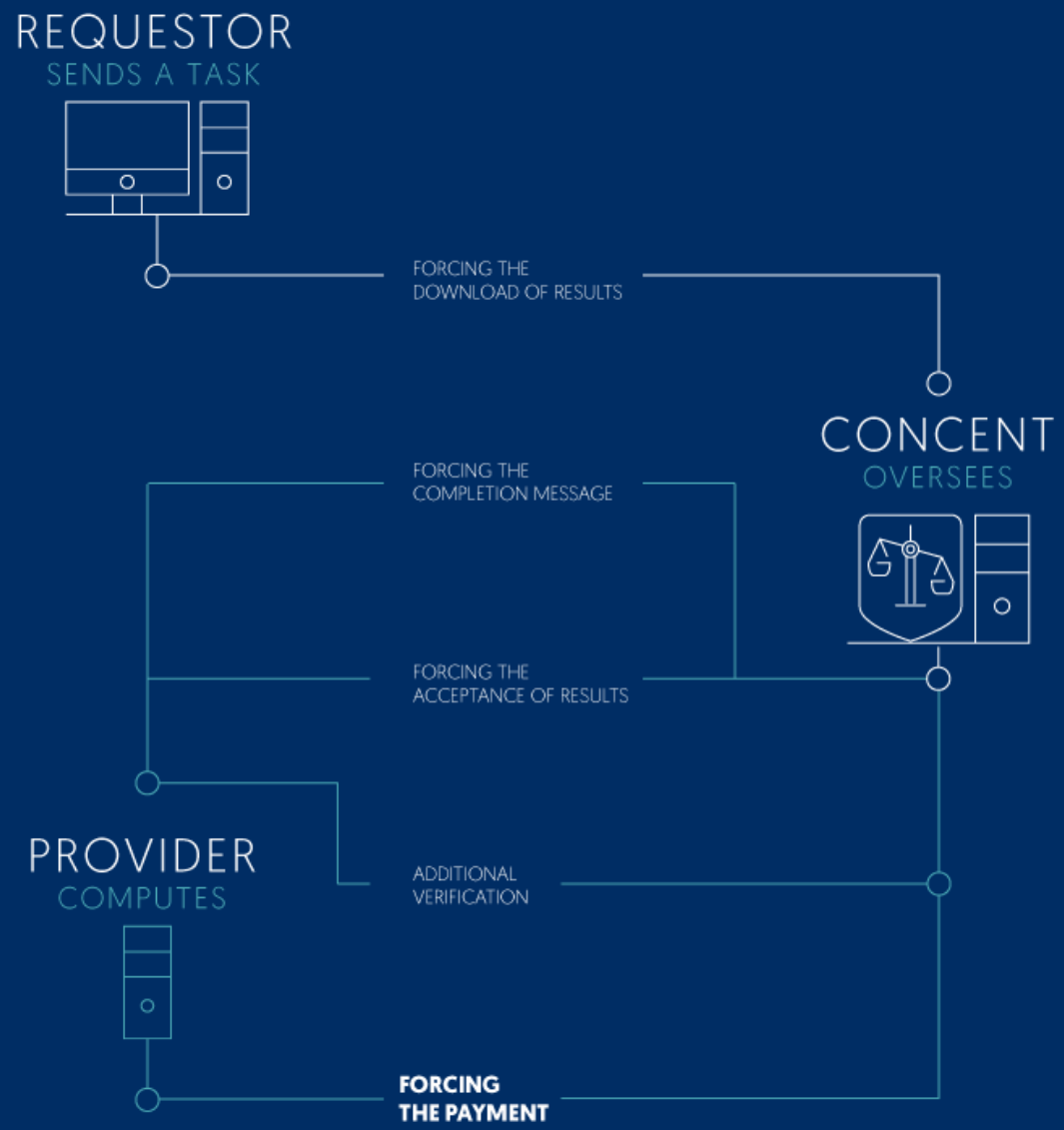


WHAT CONCENT DOES

03

Enforcing payment:

- If a Requestor does not pay on time for accepted results, then Concent will charge payments from the Requestor's deposit



HOW CONCENT WORKS

COMMUNICATION IN GOLEM

01

Requestor sends a task to a Provider

02

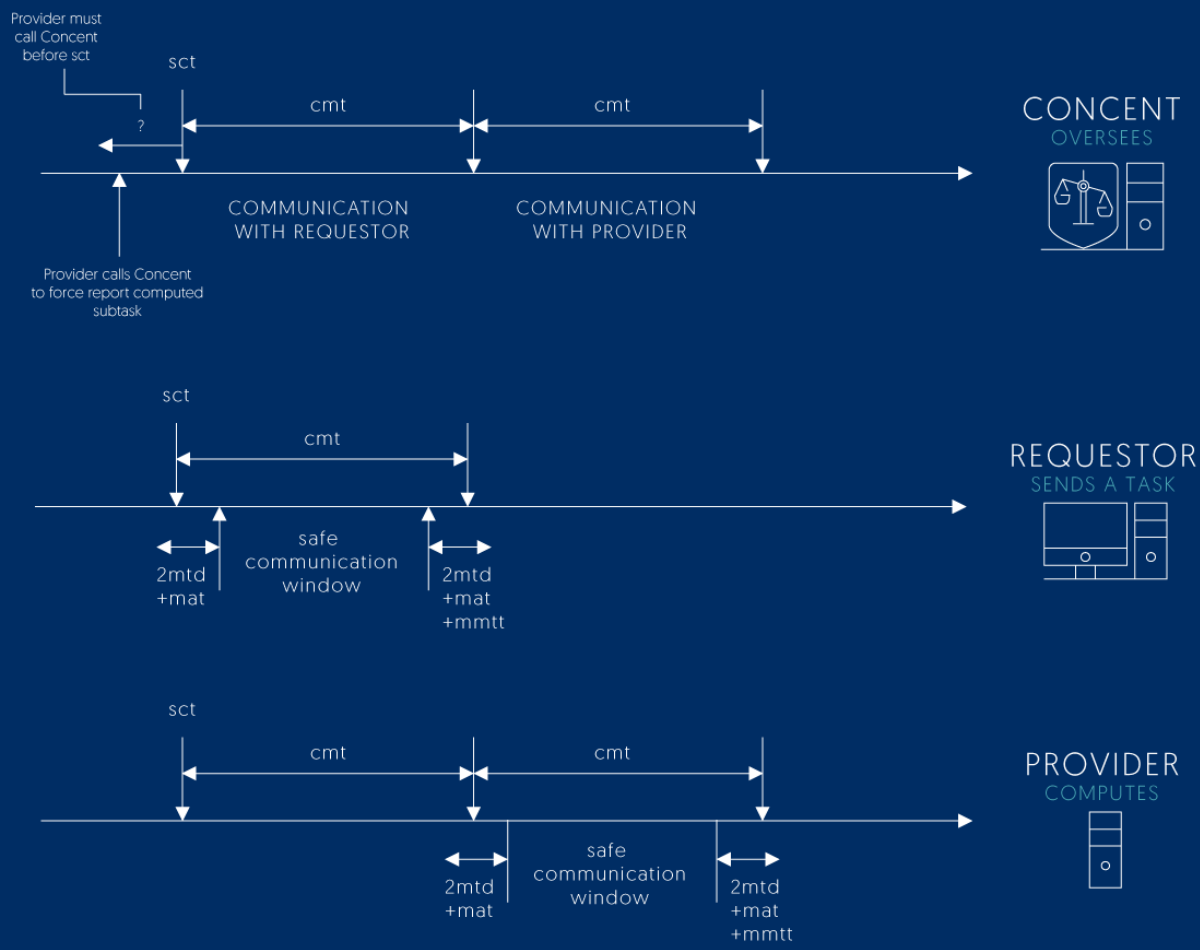
Provider calculates the task and sends the result to the Requestor

03

Requestor verifies and accepts the result

04

If the result is correct, the Requestor pays, if not they do not pay for it



SCT - SUBTASK COMPUTATION TIME
CMT - CONCENT MESSAGING TIME
MDT - MAXIMUM DOWNLOAD TIME
MAT - MAXIMUM ACTION TIME
MMTT - MAXIMUM MESSAGE TRANSPORT TIME

CONCENT'S ROLE IN SOLVING CONFLICTS

01

All Concent use-cases are meant to overcome flaws like attacks/frauds

02

Use-cases can be initialized by one or two parties

03

Use-case starts when a party does not receive a response, is not paid on time, or if their results are wrongly rejected

04

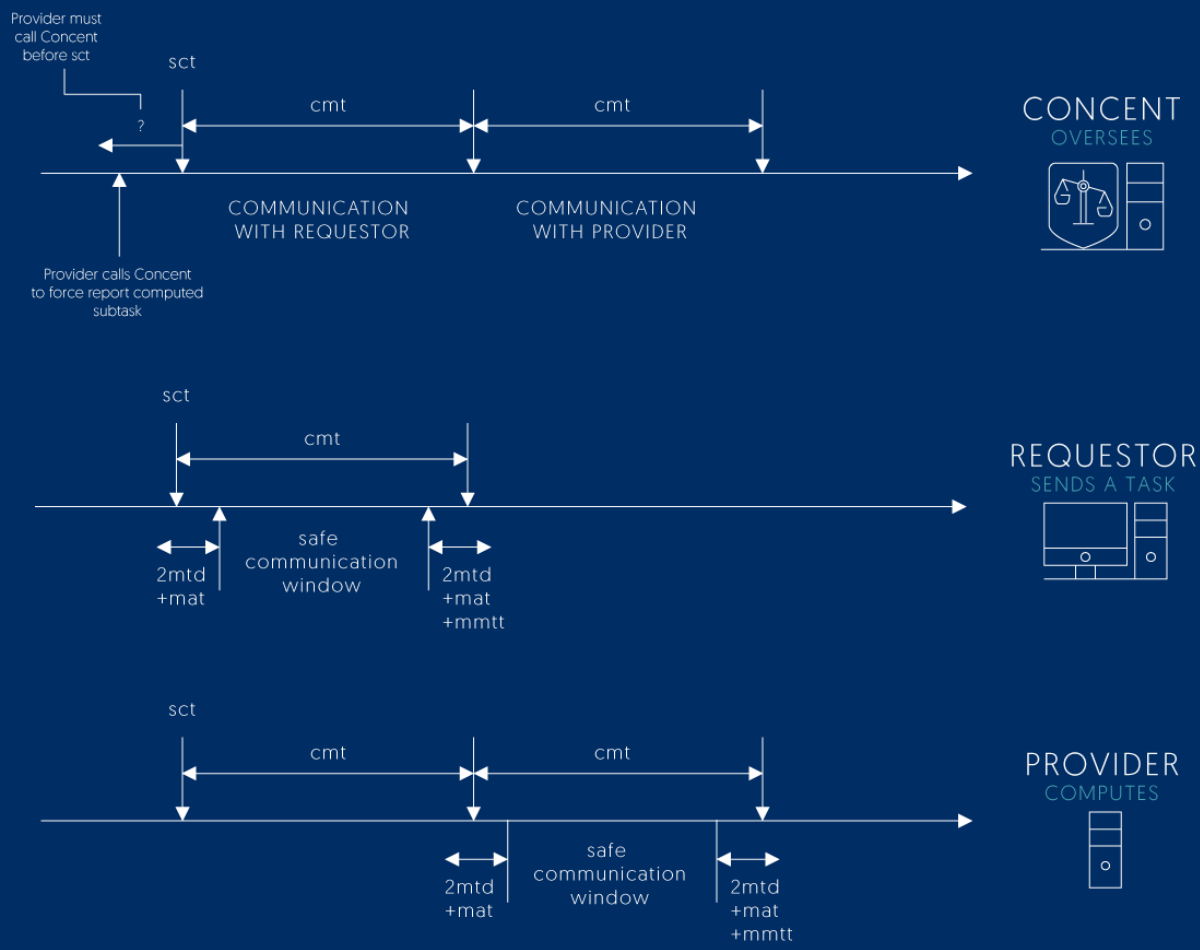
To use Concent, parties must agree to it at the beginning by putting deposits in GNT which are transferred to a dedicated Ethereum contract

05

Concent has special permissions on deposits

06

Concent is not a free service



SCT - SUBTASK COMPUTATION TIME
CMT - CONCENT MESSAGING TIME
MDT - MAXIMUM DOWNLOAD TIME
MAT - MAXIMUM ACTION TIME
MMTT - MAXIMUM MESSAGE TRANSPORT TIME

WHY CONCENT?

CAN FAIRNESS IN TRANSACTIONS BETWEEN REQUESTORS AND PROVIDERS BE ENSURED?

01

Fairness means that Requestors do not pay for incorrect results and must always pay for correct results

02

Main options for ensuring fairness in transactions:

- Use trusted third-party
- Use blockchain and/or consensus in network

03

Verification is critical for defining Concent and non-Concent solutions

IS CONCENT RELIABLE?

01

A Priori assumption: **Providers and Requestors trust Concent**

02

Under normal conditions Concent is called very rarely

03

Concent as a service is a single-point-of-failure and may be vulnerable to DDoS attacks

04

The principle is that Golem Network can calculate tasks even if Concent is down

05

Concent service will be more decentralized in the future

06

It suffices for Concent to be up 99% of time

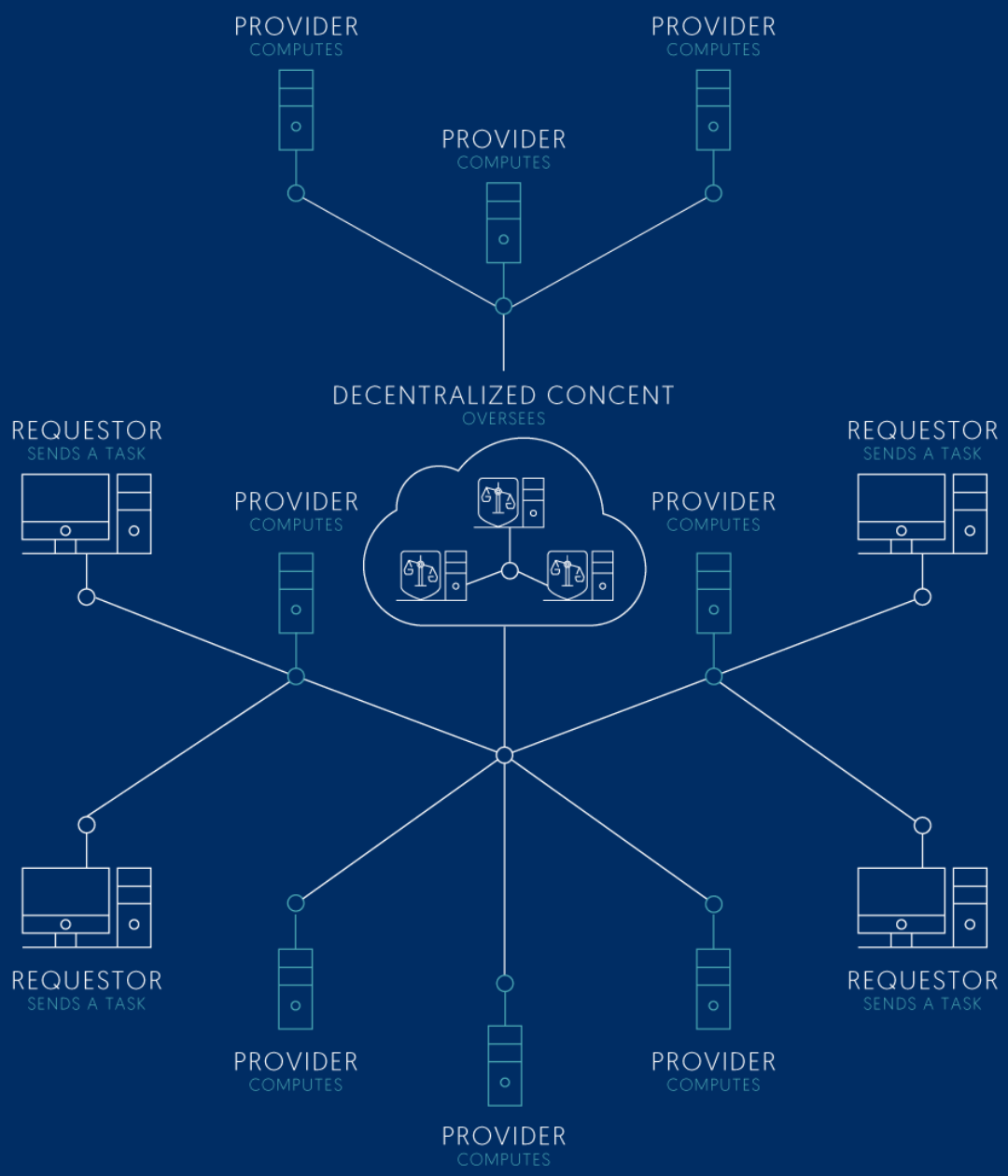
CAN CONCENT BE DECENTRALIZED?

01

In the future we plan to allow others to run valid Concent services (i.e. software developers may want to run Concent for their own applications)

02

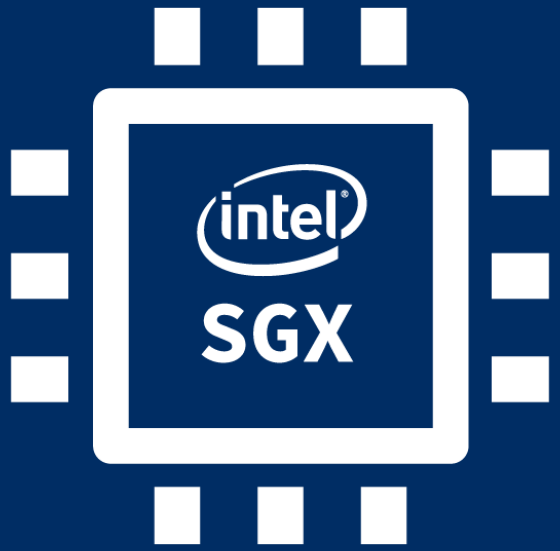
Technologies like SGX provide trustworthy and non-controlled calculations that are considered decentralized and i.e. can perform additional verification



SGX TECHNOLOGY

SGX TECHNOLOGY

> SGX is a set of CPU instructions and hardware platform enhancements that enable apps to create private areas within which code and associated data can be stored safe from compromise during execution. If used correctly, this protection can prevent compromise due to attacks from privileged software and many hardware attacks
>
> – INTEL



SGX TECHNOLOGY

01

Example usages:

- User authentication
- Sensitive data processing
- Verifiable computation

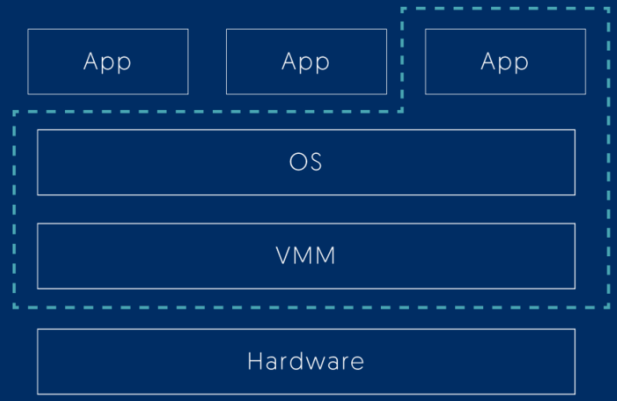
02

Verifiable computation:

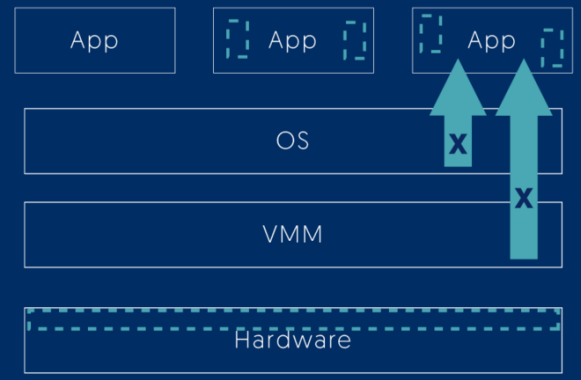
- Prevents BIOS/OS/VMM/SMM/drivers attacks
- Prevents bus snooping and memory tampering
- Provides hardware measurement mechanism
- Provides hardware attestation (local and remote)
- Provides hardware sealing (to the enclave and to the author)

– VERIFIABLE COMPUTATION IS PARTICULARLY IMPORTANT FROM GOLEM'S PERSPECTIVE

ATTACK SURFACE TODAY



ATTACK SURFACE WITH ENCLAVES

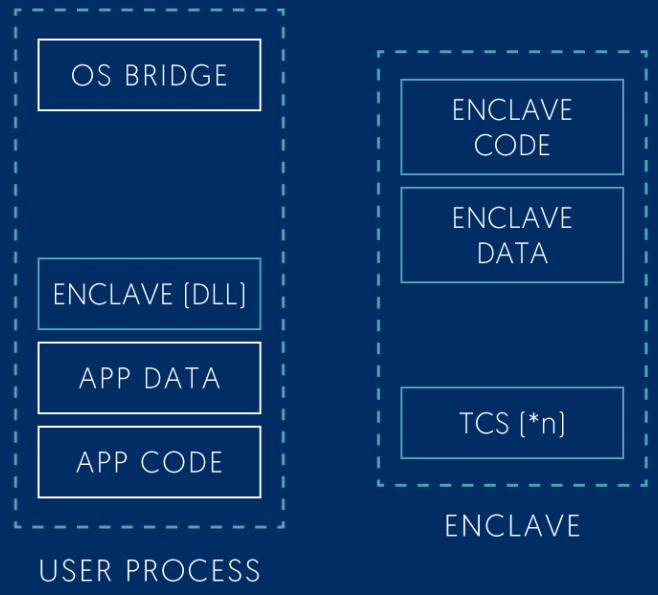


SGX TECHNOLOGY

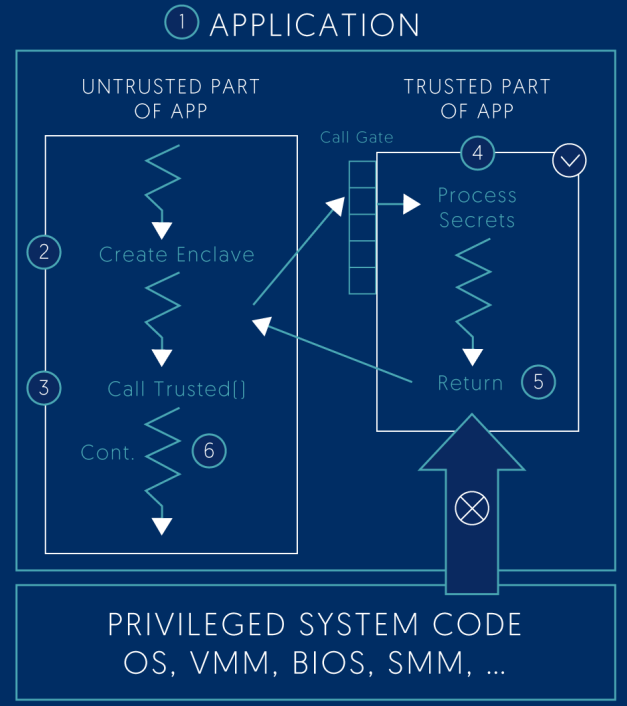
03

Enclaves:

- Similar in spirit to a DLL, SO or Dynamic Library, plus some additional configuration data
- Limited to computations only (no direct IO or OS calls allowed)
- IO or OS calls available by means of OCALLs mechanism (configured statically)
- Enclave
  - Run and hosted in an untrusted environment
  - Once instantiated it becomes a trusted part of the app (As there may appear attacks on SGX which cannot be foreseen today)



TCS – THREAD CONTROL STRUCTURE



LIFETIME OF AN ENCLAVE

01

Enclave creation:

- Binary payload (similar to a dll) which is run inside the enclave
- Configuration data:
  - Stack size
  - Heap size
  - Threads (via Thread Control Structures)
  - Enclave author's public key
  - Software version
  - Product id

02

Enclave Launch:

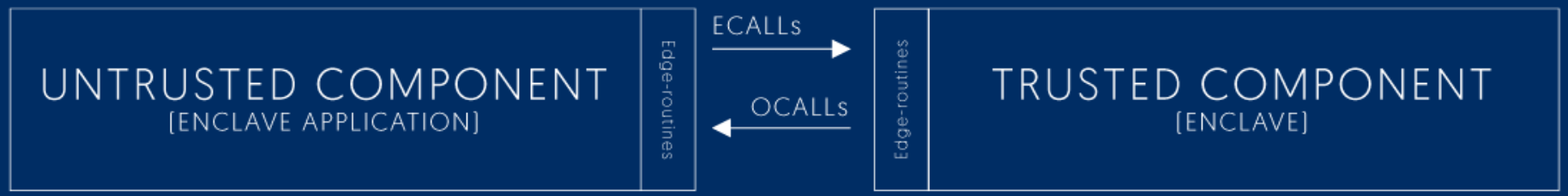
- Currently launched by Launch Enclave provided by Intel

03

Enclave Usage:

- Communication via ECALLs and OCALLs

INTEL SGX APPLICATION



ACTORS/PARTIES

01

ISV

- Enclave developer; assists in attestation; for now must be registered to Intel

02

Application

- Launches enclave and provides access to it; untrusted

03

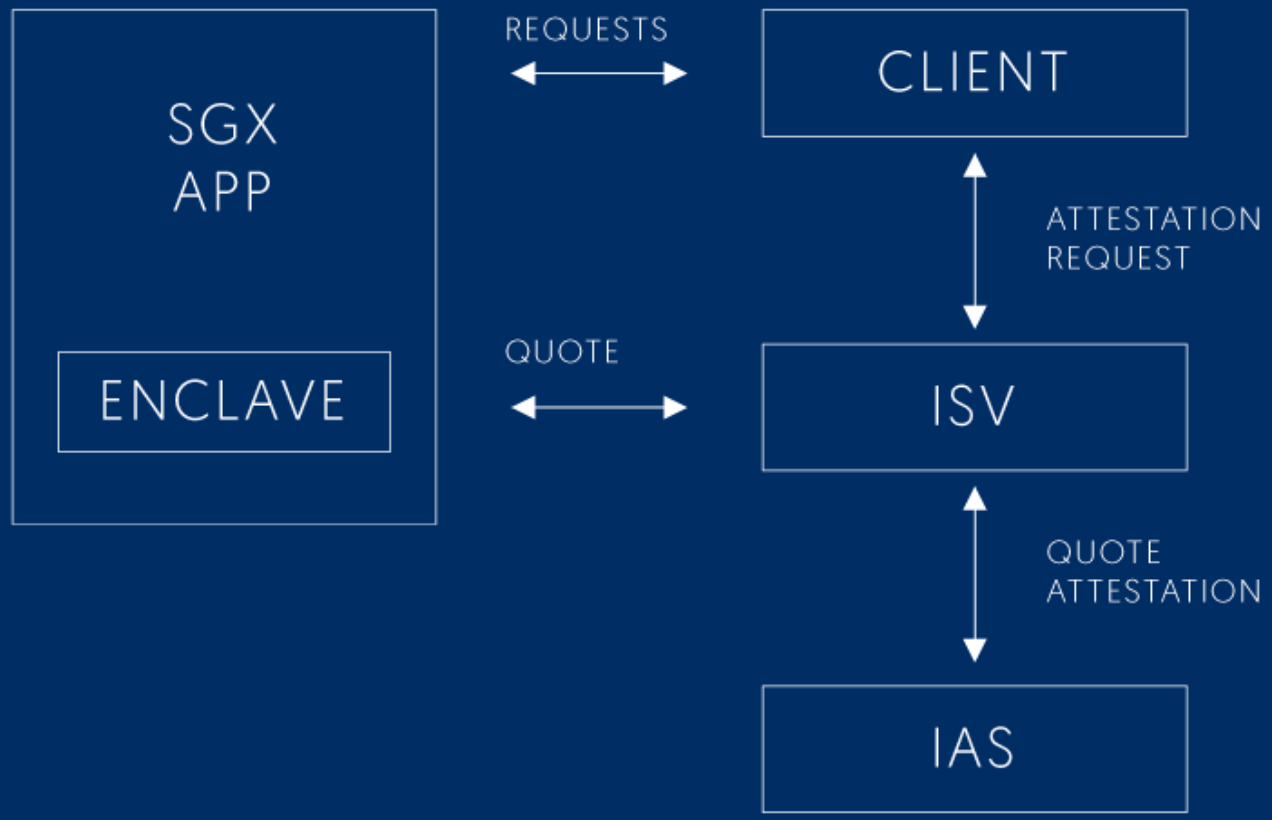
Application SGX Enclave

- Provides measurement and mrsigner for attestation and sealing; trustworthy

04

Intel IAS

- Intel Attestation Service - used to verify EPID during remote attestation



SECURITY RELATED ACTIVITIES

01

Measurement

- mrenclave – hash of vanilla enclave state including its security properties – used to prove that the enclave was instantiated correctly

02

Attestation

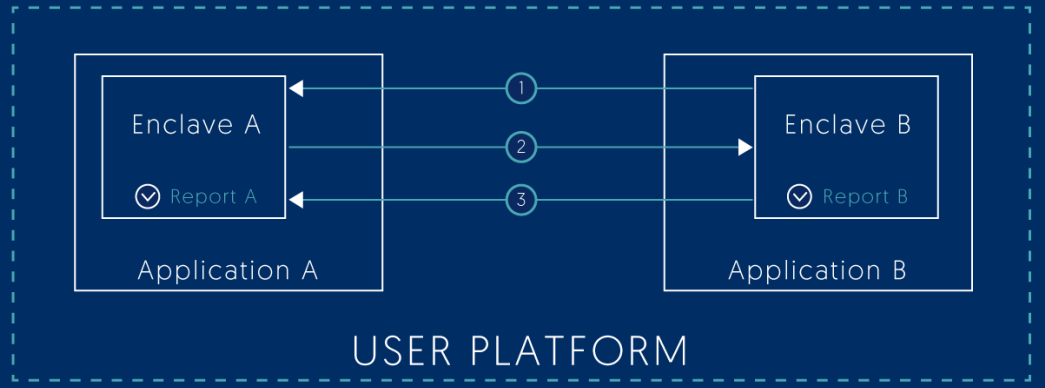
- Local – between enclaves on a single machine
- Remote – to prove the enclave validity to a remote user/client

03

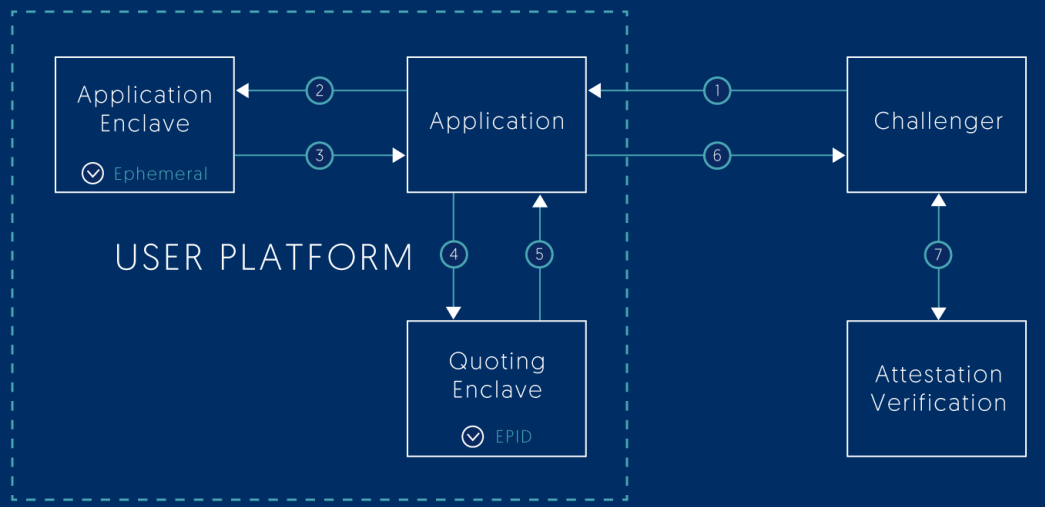
Sealing

- Used to securely store persistent data
- Seal to current enclave – only the enclave with the matching measurement can unseal the data
- To the author – used to unseal data in enclaves developed by the same author

LOCAL ATTESTATION



REMOTE ATTESTATION



CONS OF SGX TECHNOLOGY

- Intel presence is required to attest quotes and partially to launch enclaves
  - Launching an enclave with all SGX features enabled requires Launch Enclave provided by Intel
  - Currently the Intel Attestation Service (IAS) is necessary for the remote attestation to work
- ISV requires Intel approval
- Remote attestation procedure does not require IAS by design and a perfectly valid remote attestation process without IAS can be implemented (once IAS is liberated)

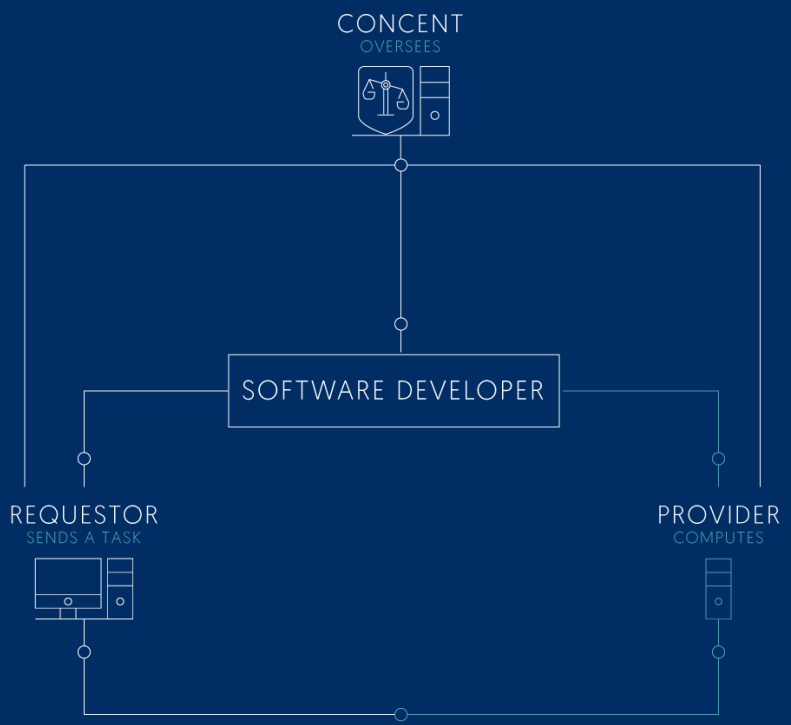
SGX AND GOLEM

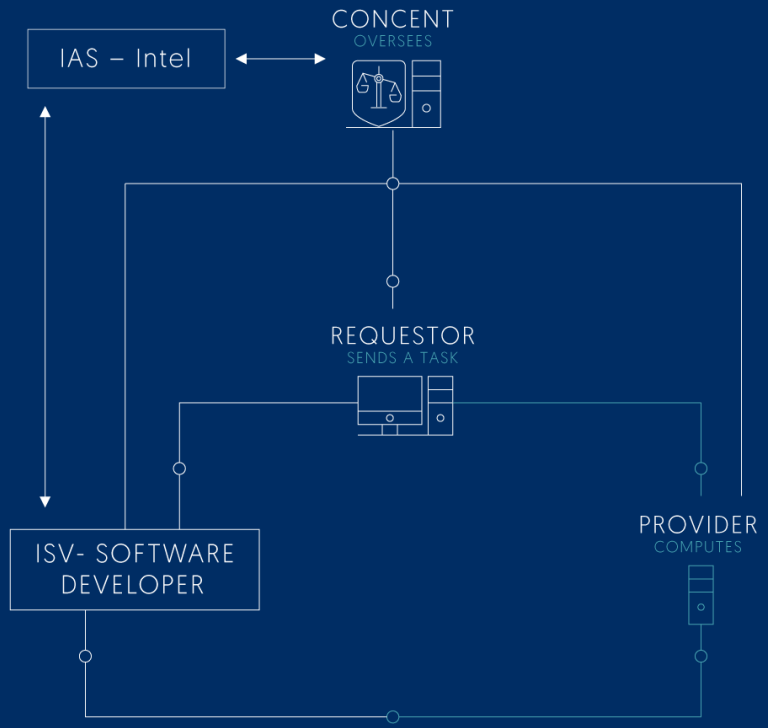
SGX AND GOLEM

01

How SGX can be used to the advantage of Golem

- Software Developer is ISV and currently can serve as attestation service - and connect to Intel to assure correctness of the enclave in question
  - Once a proof for an enclave is created by IAS, the proof can be subsequently verified offline
  - The protocol used to communicate with IAS is not critical from a security standpoint
- Requestor and Concent are "hosting" Developers code



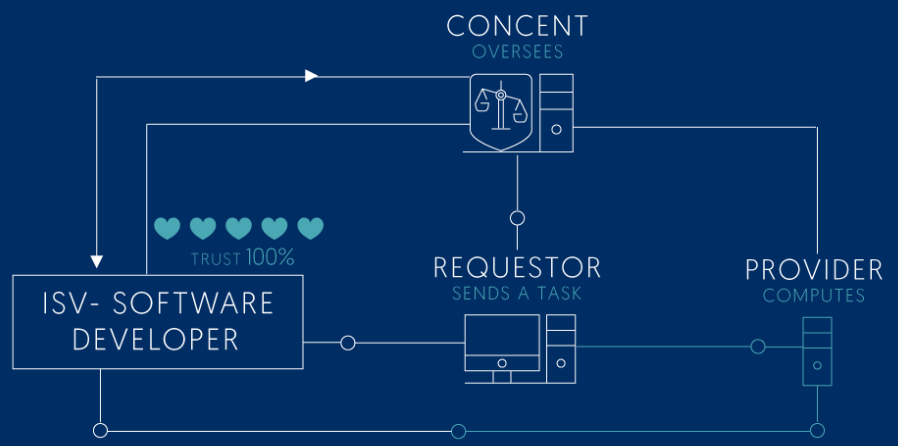


SGX AND GOLEM

02

SGX in Golem - possible future

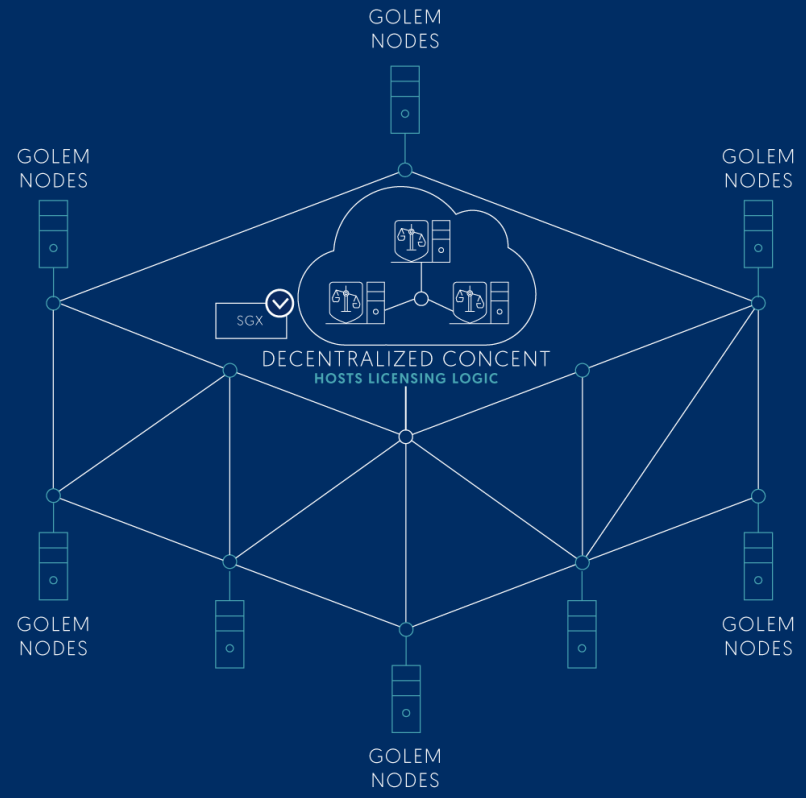
- When (if) SGX is liberated, then only ISV is required to have a high reputation (from requestors' point of view)
- ISV has skin in the game to provide a valid enclave, so they should be trusted with attesting enclaves



03

How SGX nodes can be added to Golem

- Additional Concent use cases
  - Even though SGX nodes are present and arbitrage is not as necessary as it was, additional logic related to licensing models may still be required and should be implemented in a decentralized and secure manner

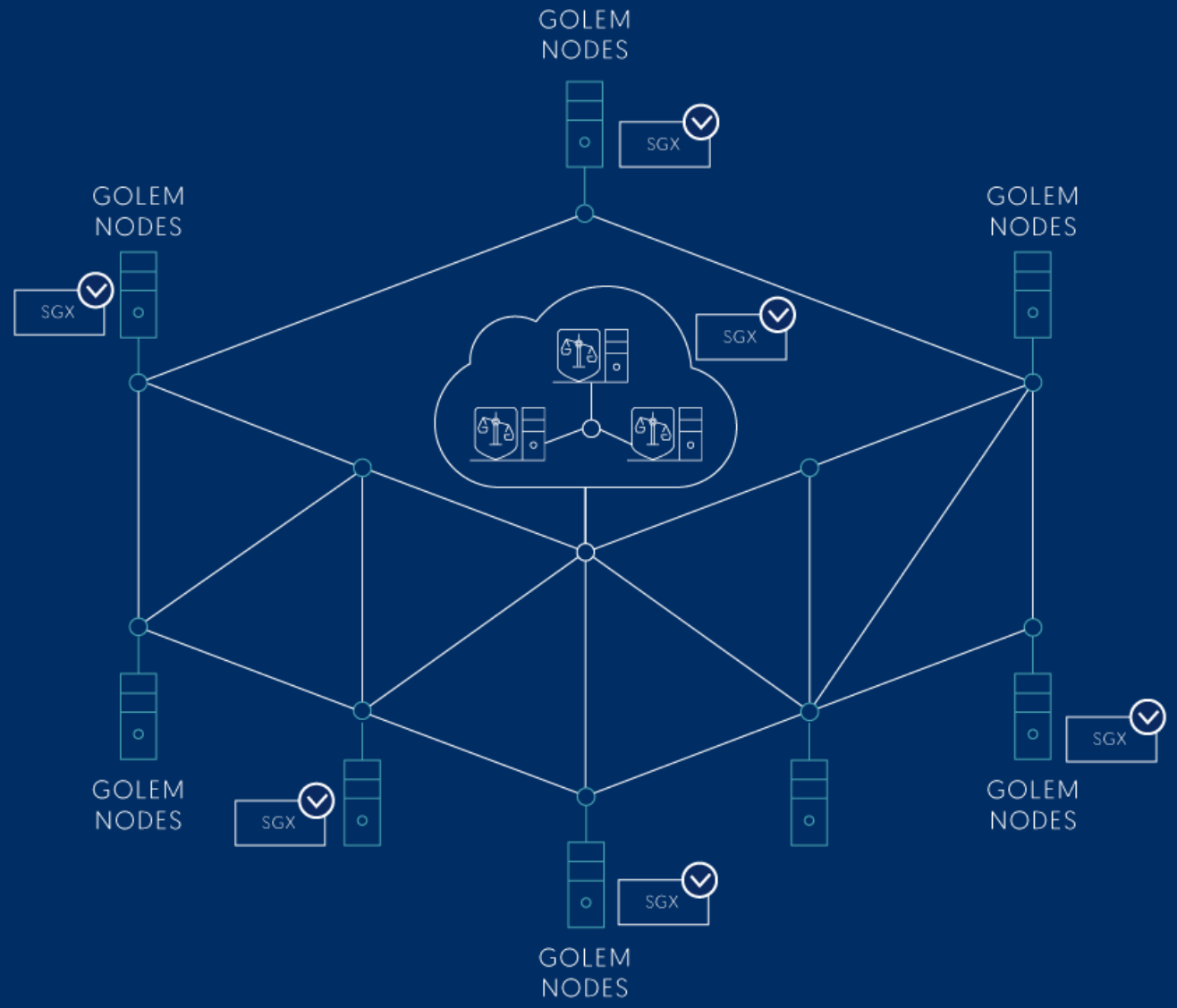


SGX AND GOLEM

03

How SGX nodes can be added to Golem

- The most resource consuming aspect of Concent is additional verification
  - Concent can delegate verification to SGX nodes increasing scalability and keeping the same level of trustworthiness. It may even happen that SGX nodes computations are considered as more trustful. This makes the most sense if SGX nodes compute noticeably slower than regular nodes
  - In such a setting a regular node may not be incentivized to call SGX nodes directly, but there would be a special case where an SGX node would be able to provably repeat computation and compare results with questionable ones
- Other solutions
  - May encompass sharing logic between one group of nodes and the state and history on other nodes

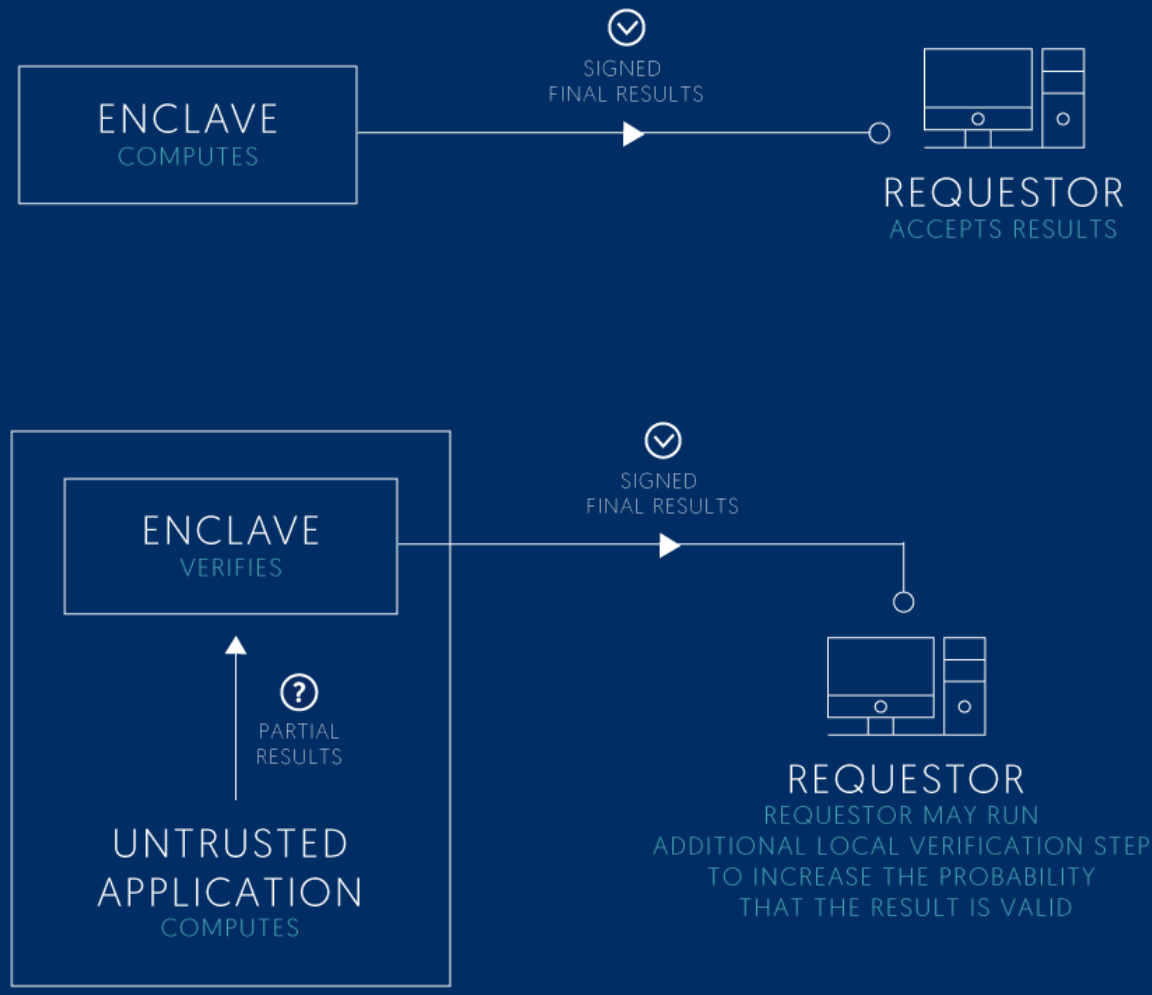


SGX AND GOLEM

04

Computations and Verification with SGX

- We can run the whole computation inside an SGX enclave (if possible) and if the result is returned - it is the result of a valid computation and no verification is needed
  - The aforementioned approach may result in less security guarantees than the default SGX enclaves, but should still be much more secure than computations carried out the regular way



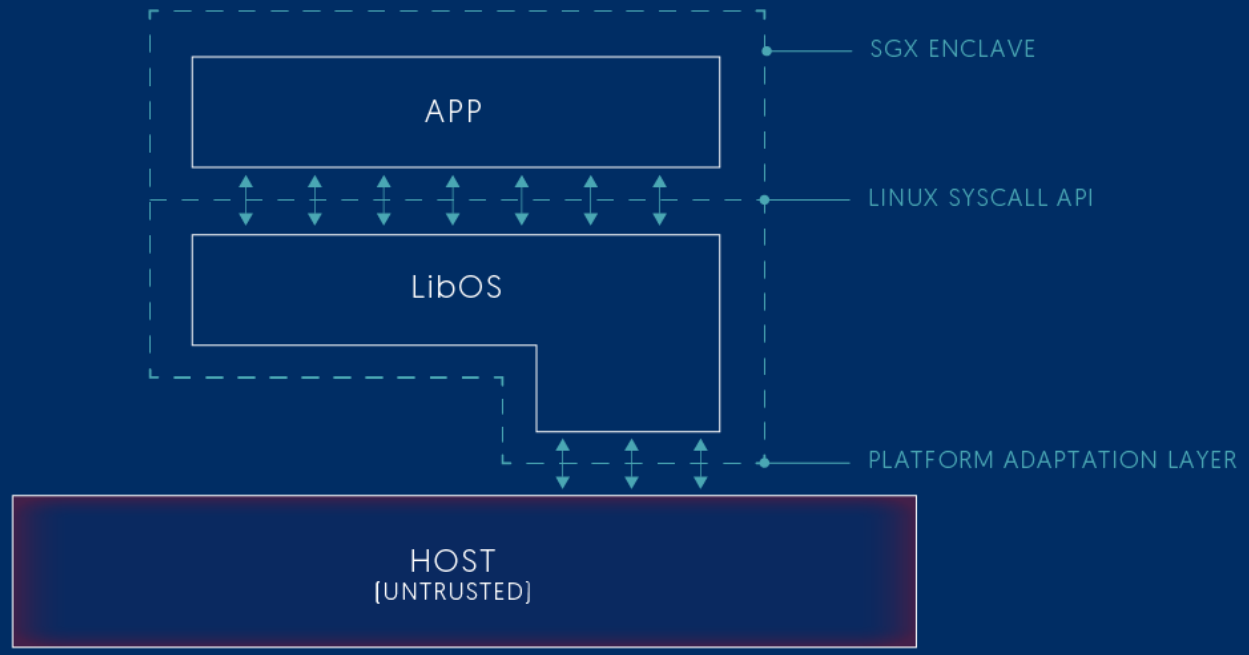
SGX AND GOLEM

04

Computations and Verification with SGX

- We can run the whole computation inside an SGX enclave (if possible) and if the result is returned - it is the result of a valid computation and no verification is needed
  - The aforementioned approach may result in less security guarantees than the default SGX enclaves, but should still be much more secure than computations carried out the regular way
  - SGX enclaves are restricted to computations only, but more general platforms allowing arbitrary computations inside the enclave are being researched and developed. We closely cooperate with Invisible Things Labs, the team/company solving problems in this exact area

GRAPHENE: EXAMPLE APPROACH FOR THE UNIVERSAL SOLUTION

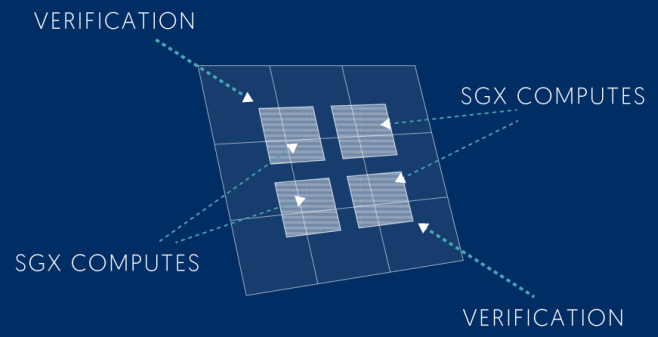


SGX AND GOLEM

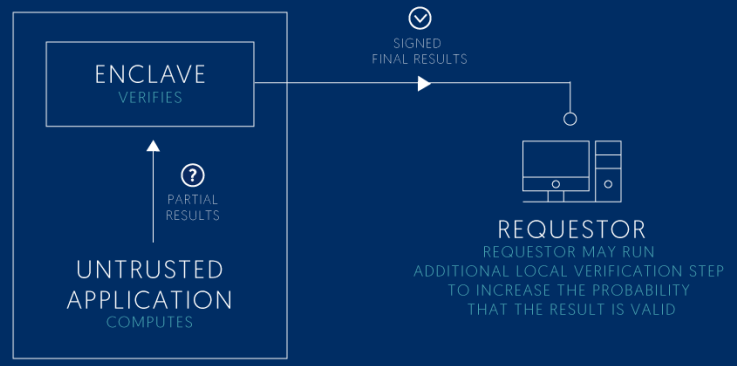
05

How SGX can help with verification

- We can use SGX nodes and redundancy to help verify (with tunable probability) the results from regular nodes
  - Just as regular redundancy, but we know that results from SGX nodes are significantly more trustworthy and can be used as reference data



- Additional verification logic can be implemented and run in an enclave whereas the main computation is performed on the regular host machine, but with intermediate results being sent to the Enclave



- Performance
  - SGX supports EPC paging (enclave's private memory protected from the host and external, physical attacks), but it may be harmful to the performance
  - Depending on the performance impact of SGX, an appropriate fraction of calculations should be delegated to SGX nodes, the less impact, the more subtasks can be sent to SGX nodes (possibly for higher price)

SGX AND GOLEM

01

Summary

- In a perfect world SGX or similar technology:
  - Should provide mechanism for verifying that declared computations indeed took place
  - Should assure privacy of sensitive data
  - Should not impose large computational overhead
  - Requestors shouldn't have to verify the results locally at all
- SGX or similar technology can be used to help decentralize server solutions in a secure manner
- The Golem network may consist of regular nodes, nodes with SGX and Concents using SGX
  - Non-SGX nodes are first class citizens but may require more mechanisms to assure valid results produced by them (e.g. local verification, redundant computations, reputation)
- Such technology seems to be a perfect match for nondeterministic computations (as defined in previous sections)

SUMMARY

SUMMARY

- There is much more to Golem than verification of broadly defined nondeterministic tasks, but this seems to be one of the most prominent challenges in a decentralized and trustless setting
- A detailed introduction to Monte Carlo rendering was presented to give you a better understanding of the problems one has to tackle when fully a decentralized infrastructure is to be prepared. As most of the issues are well known, applying them in a trustless decentralized setting requires additional thoughts in order to balance: quality, security, pricing, and efficiency
- Our approach to this challenge was presented and we shared a few ideas and directions we're already exploring in regard to improving general network security without sacrificing too much efficiency
- A few approaches to verification of rendering tasks were shown along with the introduction of Concents, optional infrastructural component purposed with securing the network and allowing software developers to implement custom licensing logic
- Then Intel SGX technology was shown as a potentially good fit with regard to both nondeterministic verification and decentralization of Concent

PIOTR JANIUK

CTO, co-founder

Golem's speaker at Devcon-0 and Devcon-1. Previously a designer and lead developer of Black Vision, a real-time rendering engine for TV broadcasting. Implemented the fastest (at the time) software jpeg2000 codec for DCP. Also took part in a few side projects related to p2p networks.

Interested in computer graphics, digital signal processing, cryptography, compilers, virtualization, blockchain tech, parallel computing, distributed computing, trustless computing and optimization techniques.

viggith@golem.network

https://www.linkedin.com/in/viggith/

THANK YOU

golem WORLDWIDE SUPERCOMPUTER

WWW.GOLEM.NETWORK

For the help with this presentation thanks to:
Substantial knowledge: Aleksandra Skrzypczak, Łukasz Gleń
Design: Jacek Muszyński, Urszula Trzaskowska (Cobaltblue)
Edit: Joanna Janiuk, Dan Horne