

Cos'è la blockchain?







- Un registro decentralizzato Che contiene:
 - Transazioni
 - Eventi
- Un algoritmo di consenso
 - Per decidere chi può scrivere e modificare il registro
- Un'po di crittografia
 - Per rendere il registro sicuro e impossibile da modificare per chi non é autorizzato a farlo
- Incentivi
 - Per incoraggiare i miner a mantenere il registro e fare in modo che tutto funzioni

**Proviamo a creare un
registro di transazioni sicuro**

Il registro

- Contiene l'informazione di chi quanto deve a chi
- Ognuno può aggiungere una nuova transazione al registro
- Facciamo il reso conto ogni settimana

Pensi di poterti fidare dei tuo amici?

-  Federica paga a Marco  300€
-  Marco paga a Cesare  5€
-  Cinzia paga a Federica  80€

Come rendere il nostro registro sicuro dalle truffe?

- Federica paga a Marco ● 300€
- Marco paga a Cesare ● 5€
- Cinzia paga a Federica ● 80€
- Federica paga a Marco ● 1000€
- Cesare paga a Marco ● 1000€
- Cinzia paga a Marco ● 1000€

Una firma digitale? v.1

- Ognuno nel registro dovrà generare una sua chiave privata e una pubblica
- Ogni transazione dovrà essere firmata dal suo mittente
- Ogni partecipante del registro verrà identificato dalla sua chiave pubblica






Firme digitali v.2

Autorizzazione(messaggio, chiavePrivata) -> Firma -> Ricovera (Firma, messaggio) -> Chiave privata

Le firme non potranno più essere copiate perché dipendono dal messaggio(dettagli della transazione).

Ora è impossibile falsificare la firma grazie alla crittografia

Come il registro attuale?

-  Federica paga a Marco  300€ Firma di federica
-  Marco paga a Cesare  5€ Firma di Marco
-  Cinzia paga a Federica  80€ Firma di Cinzia

E sicuro?

Beeeeh, no perche marco puo sempre coppiare una transazione intera







-  Federica paga a Marco  300€ Firma di federica
-  Marco paga a Cesare  5€ Firma di Marco
-  Cinzia paga a Federica  80€ Firma di Cinzia
-  Federica paga a Marco  300€ Firma di federica

Registro v.3

La risposta e 'Replay protection'

- **Da adesso ogni transazione avrà un suo numero unico(ID)**
- **Adesso quando una transazione verrà firmata con una firma digitale questa dovrà includere anche il numero della transazione**

Registro v.3

- | | | | | |
|---|--------------------------|---|------|---------------------|
|  | 1.Federica paga a Marco |  | 300€ | 1.Firma di federica |
|  | 2.Marco paga a Cesare |  | 5€ | 2.Firma di Marco |
|  | 3.Cinzia paga a Federica |  | 80€ | 3.Firma di Cinzia |

Vedete ancora alcun problema?

Altri problemi

- Le transazioni possono essere scambiate.
- Se le transazioni saranno aggiunte nel registro nello stesso tempo potrebbe crearsi un conflitto.
- La soluzione migliore sarebbe firmare tutto il registro ogni volta che una transazione viene aggiunta così la persona che vuole fare una nuova operazione nel registro in automatico è **disposta** ad accettare il passato. **Bene, però questo non sembra essere molto pratico.**

Registro v.4

 Federica paga a Marco  300€ *Firma di federica*

 Marco paga a Cesare  5€ *Firma di Marco*

 Cinzia paga a Federica  80€ *Firma di Cinzia*

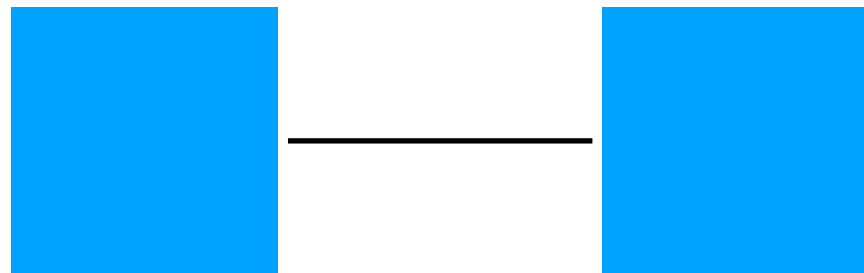
É sicuro?

Così si crea una transaction chain

- Vuol dire che ogni nuova transazione conferma la precedente contenendo anche il suo hash(quello della transazione[i - 1]).

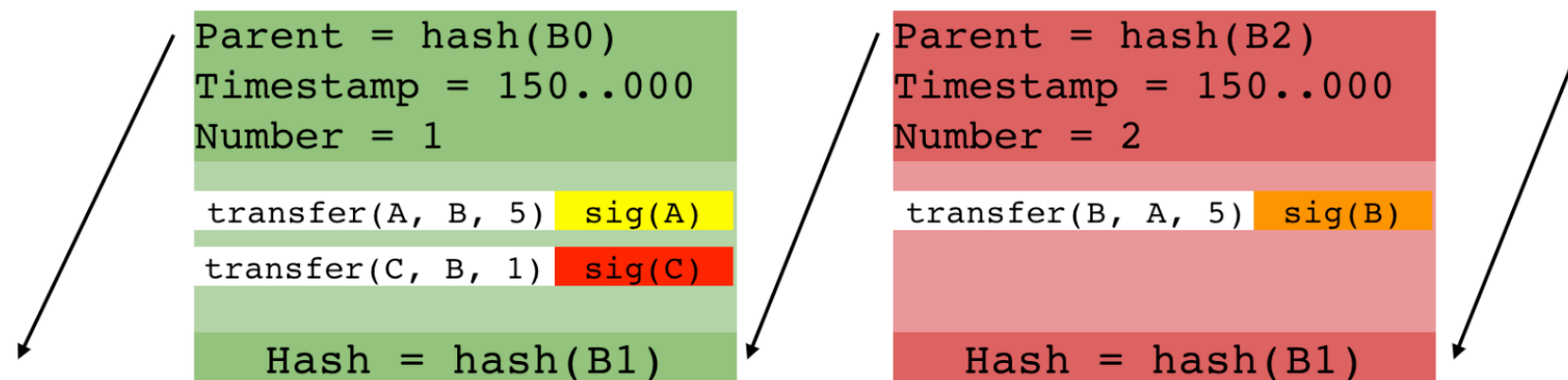
Ordinare le transazioni

- Firmare una transazione con la precedente non é la soluzione perfetta, così che le transazioni vengono ordinate in gruppi che chiamiamo blocchi.



Blockchain la definizione

- La blockchain in verità è una linked list di transazioni immutabile che è stata fatta diventare sacra.



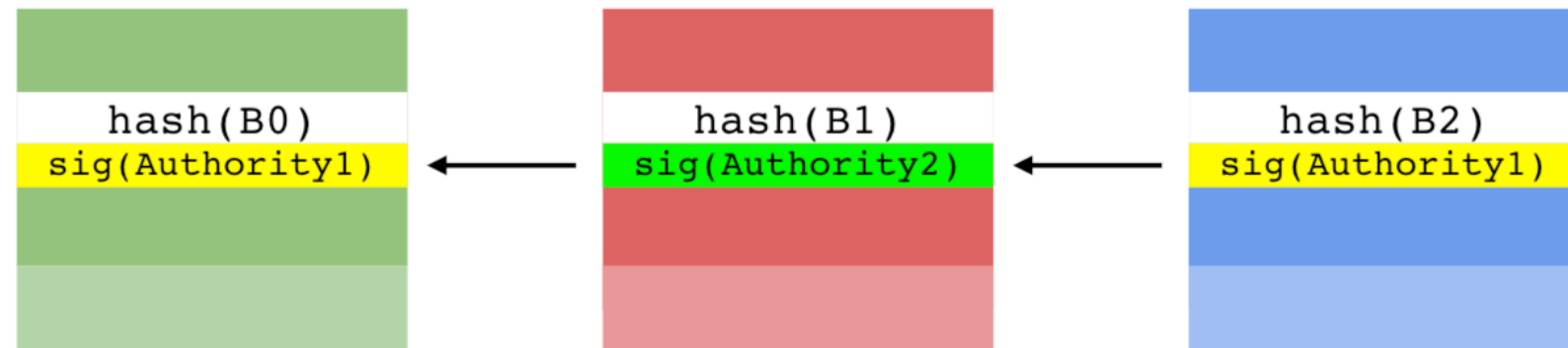
Hashes - prevent tampering (e.g. KECCAK256)

Signatures - authorize the actions (e.g. ECDSA)

Chi crea i nuovi blocchi e perché?

- Prima di creare bisogna decidere chi è autorizzato a creare nuovi blocchi.
- Forse sarebbe il caso di premiare chi crea nuovi blocchi?

L'Algoritmo di consenso



Proof of Authority

In questo caso alla blockchain vengono aggiunti solo dei blocchi che sono stati firmati da una lista predefinita di partecipanti della rete.

Algoritmo di consenso

Proof of Work

Previous block hash: <int>	
Nonce <any>	Block transactions <Object[]>
Meta <Object[]>	

Condition:

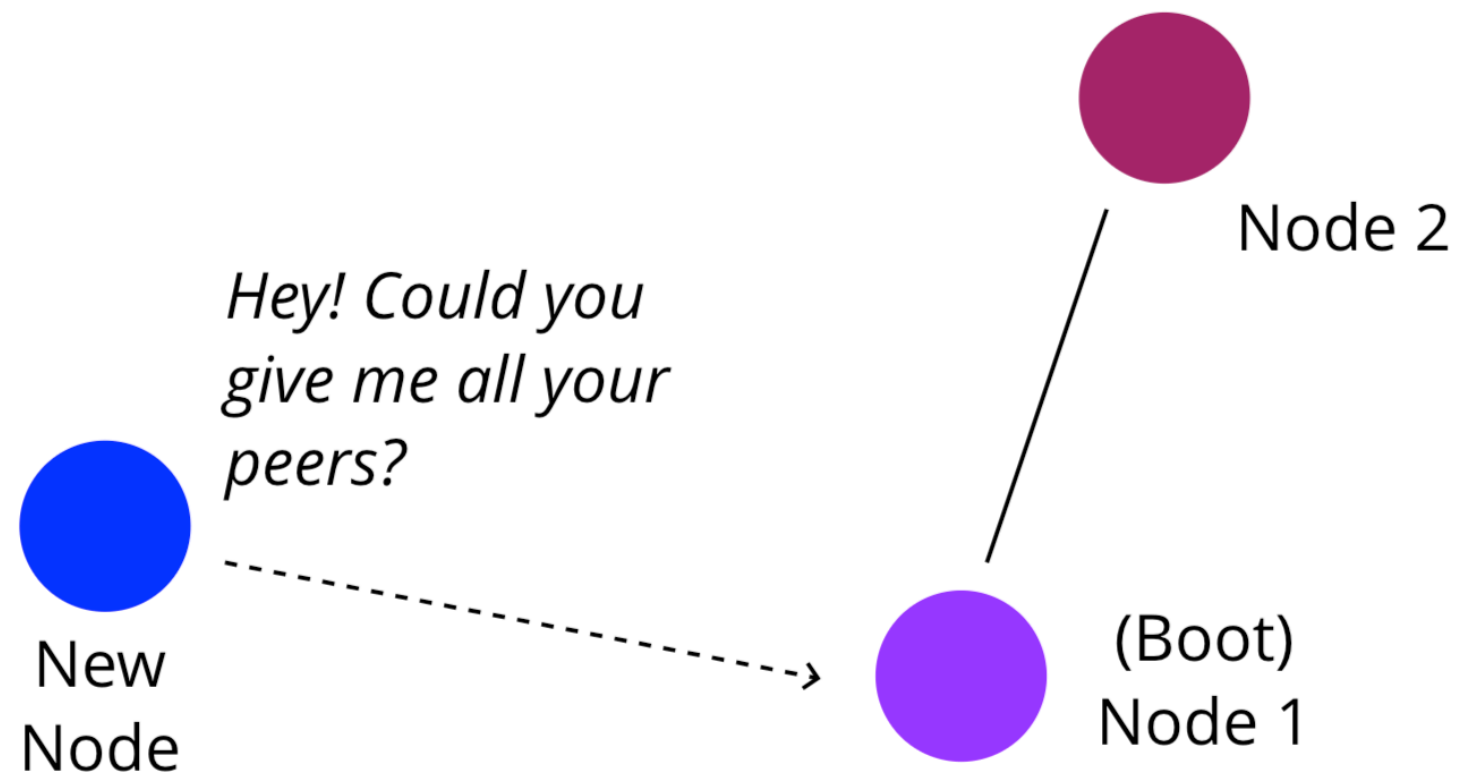
$\text{int}(\text{Blockhash}) < 2 \cdot 10^{100}$

Action:

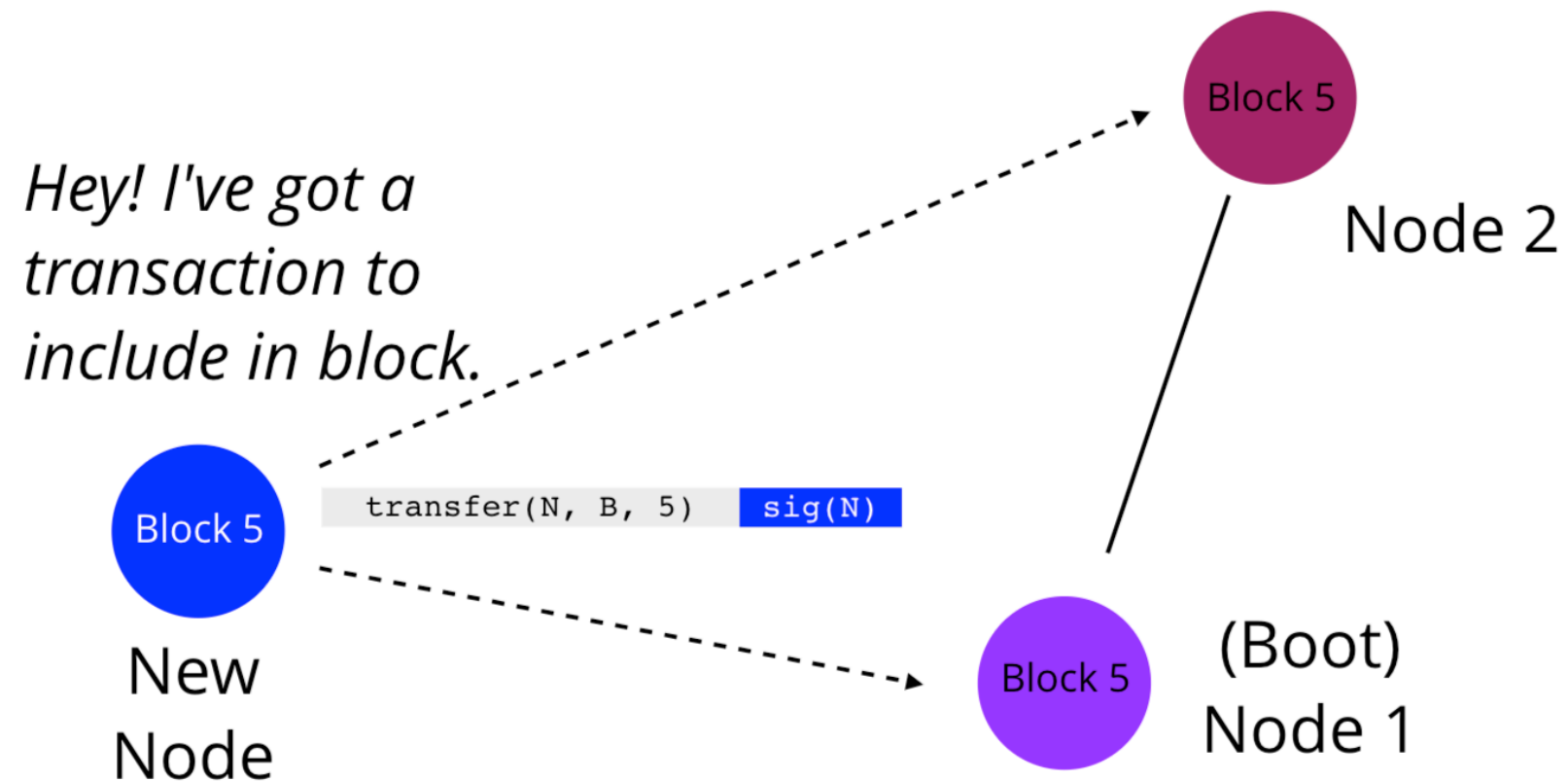
BlockHash = keccak256(
transactions,
previousBlockHash,
meta,
nonce)

```
while(!condition) {  
    nonce++;  
    blockHash = keccak256(..., nonce);  
}
```

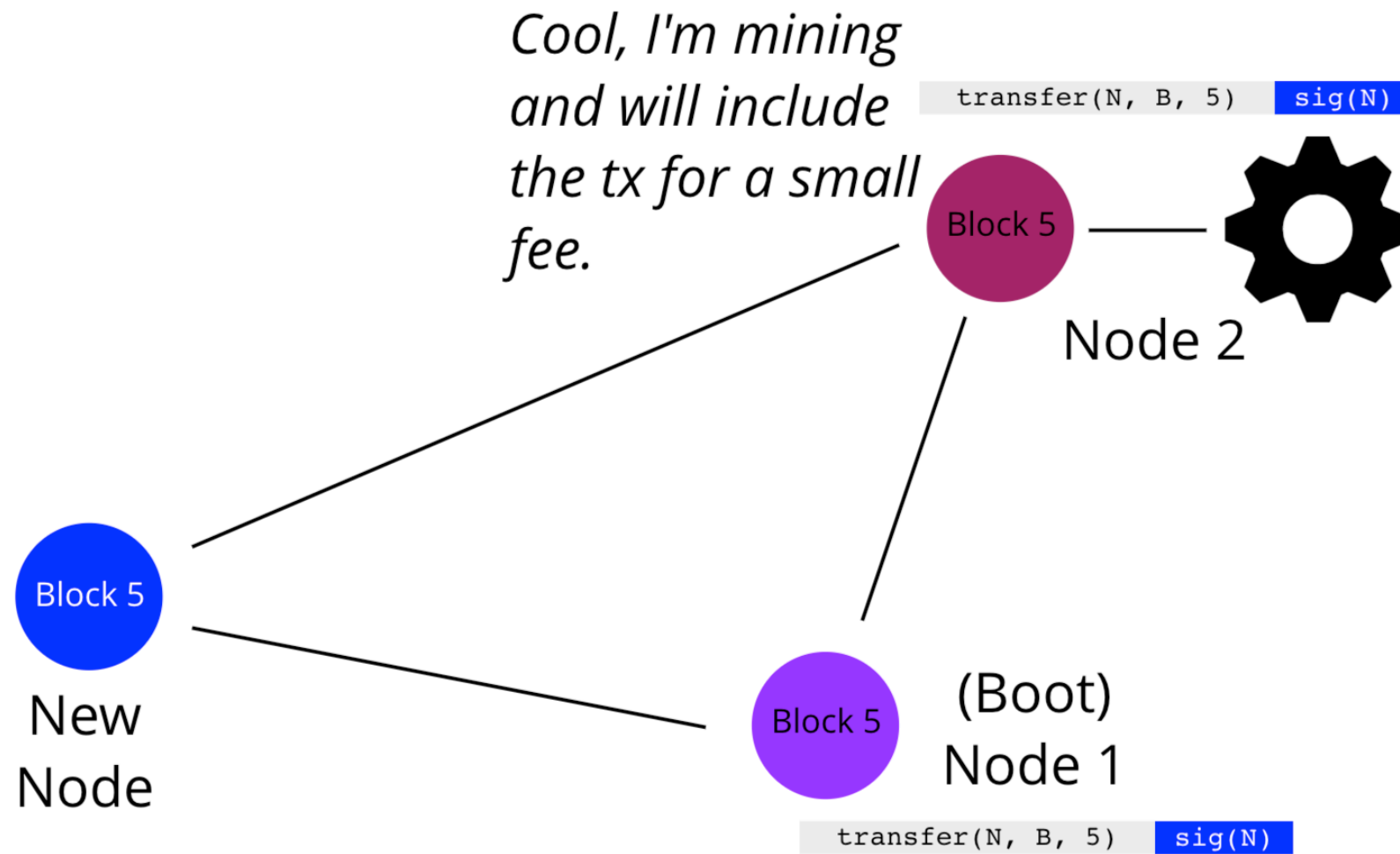
Come funziona?



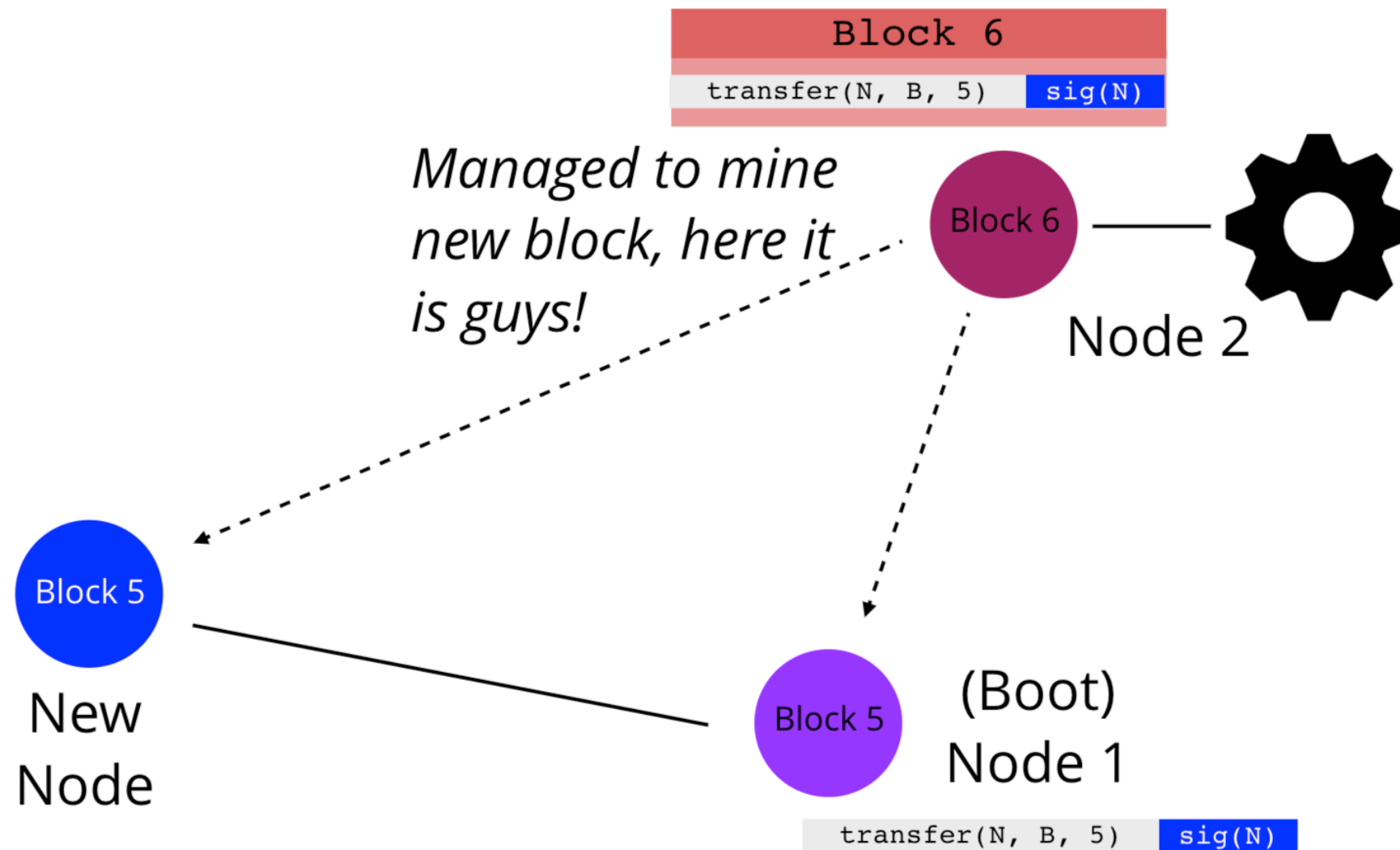
Come funziona?



Come funziona?



Come funziona?



Unpò di esempi

Bitcoin

Block Time	10 minutes
Consensus	Proof of Work - hashcash
State	Account Balances of BTC / UTXO*
Transactions	Value Transfers*
Launched	2009
Block Reward	12BTC (halving) ~ 21M total coins

Ethereum

Block Time	14 seconds
Consensus	Proof of Work - ethash*
State	Arbitrary
Transactions	Turing-complete / programmable
Launched	2015
Block Reward	3.75ETH (+uncles) ~ Unlimited coins

Come funziona Ethereum?

Quando si parla dei costi di Ethereum bisogna ricordarsi di:

- 1. Gas**
- 2. GasPrice**
- 3. Wei**

Gas

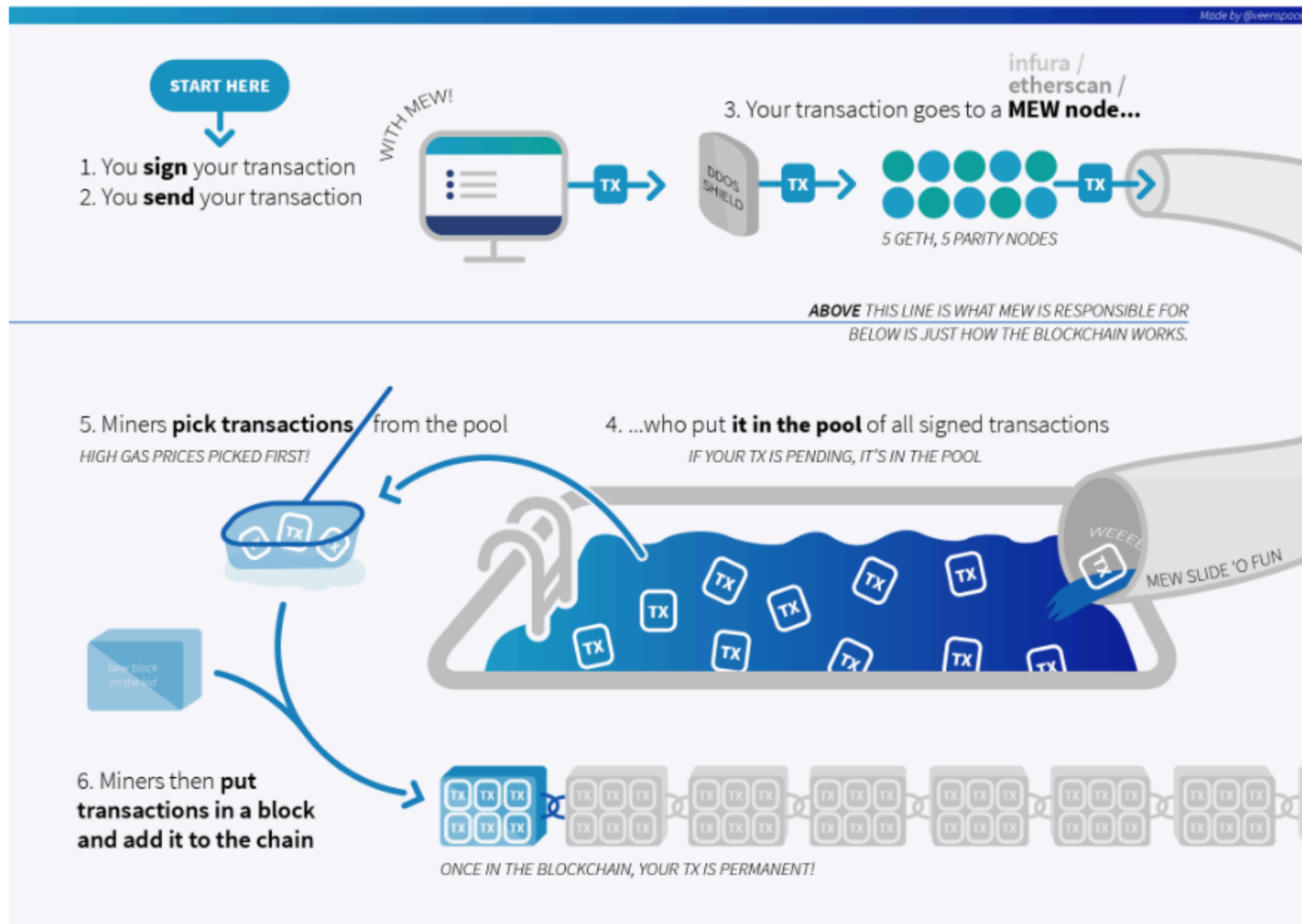
Nell'ecosistema dell'Ethereum il termine **Gas** descrive quanto costa eseguire dell'opcode sulla blockchain.

1. Sulla blockchain Ethereum per eseguire la funzione keccak256 bisogna subire il costo di 30gas + 6 gas per ogni 256bit di dati che vengono hash'atti.
2. Perciò il gas in verità é una transaction fee perché ogni azione sulla blockchain é una transazione.

Gas Price



MyEtherWallet Behind-The-Scenes



Questa infografica ci fa capire che il gas price in verità è il costo medio definito in gas delle ultime transazioni che sono state messe nel pool e nella blockchain

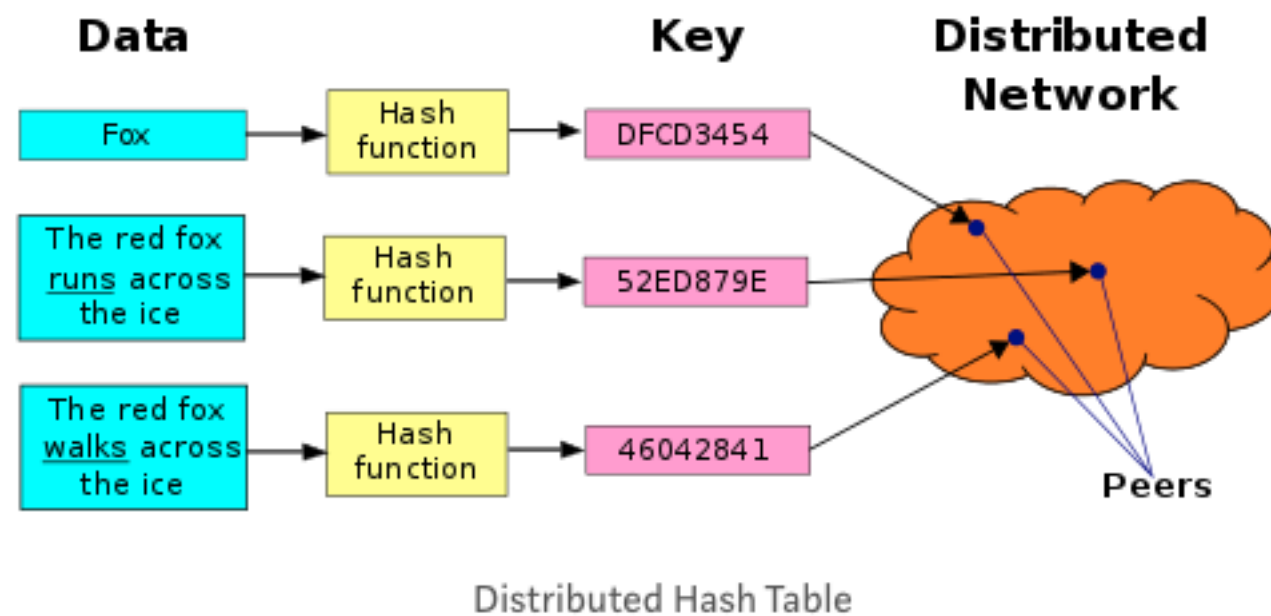
IPFS

<https://ipfs.io/>



Cos'è?

IPFS é una rete globale P2P che permette ai suoi utenti di hostare e scaricare file. Nel caso del IPFS ognuno può diventare un nodo, ma non tutti i nodi sono costretti a mantenere i stessi dati.



Come usarlo?

1.Download

// MacOS

wget https://dist.ipfs.io/go-ipfs/v0.4.18/go-ipfs_v0.4.18_darwin-386.tar.gz

// Linux

Wget https://dist.ipfs.io/go-ipfs/v0.4.18/go-ipfs_v0.4.18_linux-386.tar.gz

2.Installazione

tar xvfz go-ipfs.tar.gz

cd go-ipfs

./install.sh

3.Diventare online

ipfs daemon

4.Scaricare gattini

ipfs cat /ipfs/QmW2WQi7j6c7UgJTarActp7tDNikE4B2qXtFCfLPdsgaTQ/cat.jpg >cat.jpg

Open cat.jpg

5. Web UI

<http://localhost:5001/webui>

Come usarlo con Python?

Pip3 install ipfsapi

Python3

```
>>> api = ipfsapi.connect("127.0.0.1", 5001)
```

```
>>> res = api.add("my_file.file")
```

```
>>> res
```

```
{'Hash': 'QmWxS5aNTFEc9XbMX1ASvLET1zrqEaTssqt33rVZQCQb22', 'Name': 'my_file.file'}
```

```
>>> api.cat("/ipfs/QmW2WQi7j6c7UgJTArActp7tDNikE4B2qXtFCfLPdsgaTQ/cat.jpg")
```


Swarm

<https://swarm-guide.readthedocs.io/en/latest/introduction.html>



swarm

Whisper

Cos'è?

Whisper è un protocollo di comunicazione offchain P2P che in verità è un layer sopra il protocollo su cui è basata tutta la rete Ethereum cioè $\text{E}\text{V}\text{p}2\text{p}$ Wire Protocol.

Come funziona esempi:

Una buona analogia per capire come funziona whisper è la radio. Nel caso della radio per ricevere un'informazione bisogna conoscere la frequenza su cui i dati sono radiodiffondenti. Lo stesso su whisper.

Esempio: Whisper message:

```
{topic: 'anyStringHere', message: 'Hello World'}
```

Ora per riceverlo basta conoscere il suo topic

Come usarlo

- Whisper essendo un protocollo offchain non richiede nessun codice in solidity perciò e suo essere eseguito su ogni PC con geth acceso.
- Può essere usato direttamente da web3 pero la libreria che vi consiglio è embark di status:
https://embark.status.im/docs/messages_javascript.html

Esempio

listening to messages

```
EmbarkJS.Messages.listenTo({topic: ["topic1", "topic2"]}).then(function(message) {  
  console.log("received: " + message);  
})
```

sending messages

You can send plain text

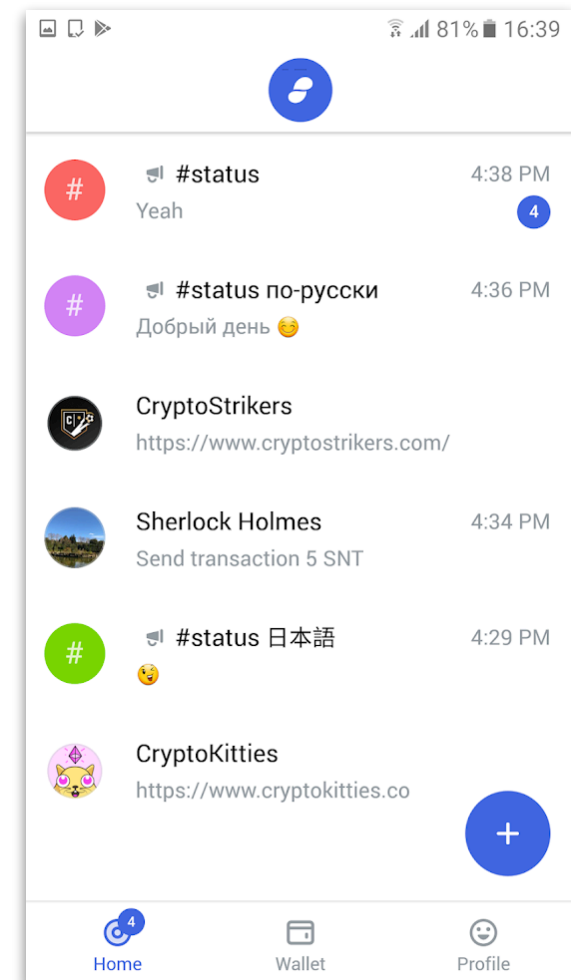
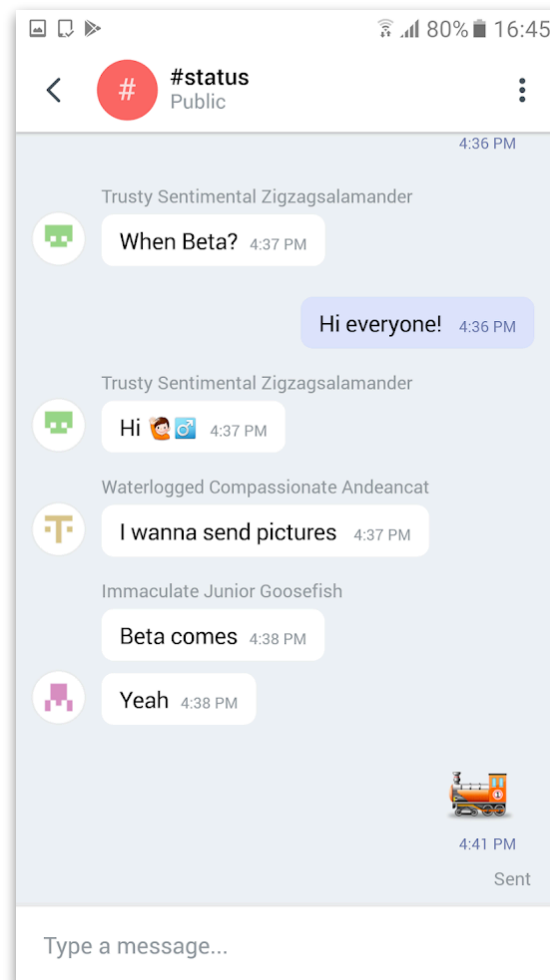
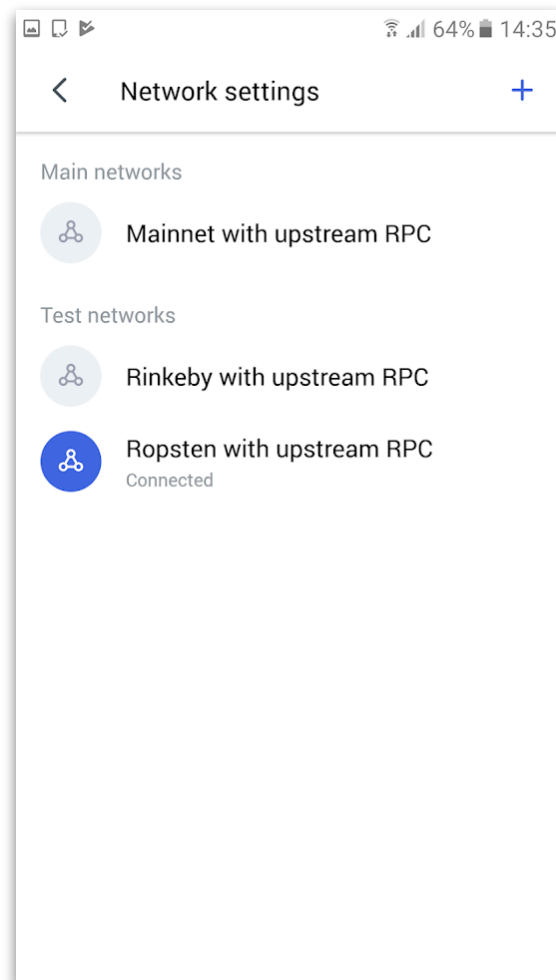
```
EmbarkJS.Messages.sendMessage({topic: "sometopic", data: 'hello world'})
```

Or an object

```
EmbarkJS.Messages.sendMessage({topic: "sometopic", data: {msg: 'hello world'}})
```

Example use-case

Status Chat



Ethereum networks:

Tipi

Pubbliche:

E un tipo di network dove ognuno può leggere il suo stato e fare transazioni ed aspettarsi che la sua transazione venga inclusa alla blockchain come valida.

Questa blockchain è del tutto decentralizzata e aperta.

Esempi:

- Ethereum Mainnet (<https://etherscan.io/txs>)

- Ethereum Ropsten testnet (<https://ropsten.etherscan.io/>)

- Ethereum Rinkeby testnet (<https://rinkeby.etherscan.io/>)

Consorzio:

E una network (di solito proof of authority) dove ce un numero n di nodi di cui solo un numero x può validare i nuovi blocchi.

Questo tipo di network è spesso usato insieme a una API centralizzata.

Queste caratteristiche rendono questa network semi-centralizzata.

Privata:

Questo tipo di network è del tutto privata e centralizzata solo una organizzazione centrale può creare nuovi blocchi e cambiare lo stato della blockchain.

Disclaimer: Tutte queste blockchain possono essere lette da chiunque è connesso alla network.

<http://www.ethdocs.org/en/latest/network/connecting-to-the-network.html#public-private-and-consortium-blockchains>

Setup

1. Creare un node Ganache or Geth
2. Installare npm o yarn, truffle, solc
3. Start a node

1.Node.Download

Ganache

Linux:

```
wget https://github.com/trufflesuite/ganache/releases/download/v1.2.2/ganache-1.2.2-x86_64.AppImage  
chmod a+x ganache-1.2.2.AppImage  
./ganache-1.2.2.AppImage
```

Geth

MacOS:

```
brew tap ethereum/ethereum  
brew install ethereum
```

Linux:

```
sudo add-apt repository -y ppa:ethereum/ethereum  
sudo apt-get update  
sudo apt-get install ethereum
```

Truffle

```
npm install -g truffle
```

Solc

```
brew install solidity  
sudo apt-get install solc  
npm install -g solc
```

Start node

```
geth  
Oppure  
./ganache.ApplImage
```