Block Valley Documentation

Versió 0.1.0

BTCA Labs

10 d'oct., 2019

Continguts:

1 Què és Block Valley? **1**

2 Les xarxes **3**

2.1 Xarxa permisionada *Ethereum* 3

2.1.1 Paràmetres de la xarxa 3

2.1.2 Funcionament de la xarxa 8

2.1.3 Visió global del funcionament 10

2.1.4 Ús de la xarxa 11

2.1.5 Governança 20

3 Indexs i taules **23**

CAPÍTOL 1

Què és Block Valley?

Block Valley és un consorci promogut per empreses andorranes que treballen amb tecnologia *blockchain*. L'objectiu del consorci és promoure la tecnologia *blockchain* al país, creant xarxes permisionades en col·laboració de tots els integrants del consorci, en benefici del país.

CAPÍTOL 2

Les xarxes

Actualment, estem creant la primera d'aquestes xarxes. Aquesta xarxa, usarà *Ethereum* com a plataforma. A la taula de continguts hi trobareu més informació sobre aquesta.

2.1 Xarxa permisionada *Ethereum*

Donada la gran popularitat de la plataforma de *Ethereum*, hem començat per formar una xarxa permisionada usant aquesta.

2.1.1 Paràmetres de la xarxa

Per a formar la xarxa permisionada, però, hem hagut de escollir uns paràmetres diferents de la xarxa principal d'*Ethereum*. Bàsicament, els canvis d'aquests paràmetres són els següents:

Paràmetre	Xarxa principal	Block Valley
Consens	Proof of Work Algorisme *Ethash*	Proof of Authority Implementació Clique
Periode de blocs	15 segons Flux constant	0 segons Només quan hi ha transaccions
Identificador de la xarxa	1	2383 Xarxa de proves *enclar*
Portes d'accés a la xarxa	*Hard-coded* als clients de *Ethereum*	Definits als fitxers de configuració (i a les imatges de *Docker*)
Moneda	Ethers Generació en cada bloc	Block Valleys Andorrans (BVA) Generació al gènesis

Als següents capítols de la documentació, detallarem la decisió dels paràmetres de la xarxa **Block Valley**, comparant-los amb la xarxa principal.

Consens

El mecanisme de consens en una *blockchain* és el mecanisme que determina què és requereix per afegir un bloc nou a la *blockchain*. Aquest mecanisme ha d'assegurar que tots els nodes de la xarxa puguin coincidir de forma independent quina és la *blockchain* vàlida partint del primer bloc o bloc *gènesis*, les regles de consens i el mecanisme de consens.

Escollint el mecanisme de consens

Proof of Work

A les xarxes *blockchain* públiques, el mecanisme de consens més conegut és la prova de treball o **Proof of Work** (PoW). Aquest mecanisme de consens permet que tothom pugui participar en l'addició de nous blocs a la *blockchain*.

Per a poder afegir un nou bloc a la *blockchain*, caldrà realitzar còmput intensiu a fi de trobar la solució a un problema determinat (trobar un *hash* dins d'un llindar determinat). Aquest còmput intensiu es veu recompensat a través de les comisions de les transaccions del bloc afegit i/o noves unitats de moneda.

Les entitats que dediquen recursos computacionals per afegir nous blocs a una *blockchain* s'anomenen miners o en anglès *miners*

Aquest mecanisme permet que sense tenir identitats, es pugui decidir de forma decentralitzada quin és el següent bloc i per tant, en conjunt, quina és la *blockchain* vàlida.

Nota: A més, els atacants que vulguin censurar transaccions éssent els únics que tenen la potestat d'afegir nous blocs a la cadena, hauran de tenir una quantitat de recursos excessiva en comparació amb la resta de miners.

Proof of Authority

A les xarxes permisionades, usar la prova de treball o **Proof of Work** (PoW) per a decidir qui serà l'encarregat de crear el següent bloc de transaccions, no és aconsellable per la seves conseqüències en la seguretat d'aquesta.

Bàsicament, un atacant amb recursos suficients podria reescriure tota la *blockchain*. Si el que volem és que només un cert grup de participants tinguin aquesta potestat d'afegir blocs a fi d'evitar aquest i altres atacs, hem d'escollir un altre mecanisme de consens.

Per això, usarem com a mecanisme de consens la prova d'autoritat o **Proof of Authority** (PoA), on un conjunt d'identitats seleccionades, seràn qui podràn afegir blocs a la cadena.

Proof of Authority

A *Proof of Authority*, en comptes de cercar la solució a un problema que requereix de computació intensa, requerim la *signatura digital* d'una de les identitats aprovades per afegir un nou bloc a la cadena.

D'aquesta manera només les entitats seleccionades podran afegir blocs a la cadena.

Les entitats que afegeixen blocs a la cadena en un *Proof of Authority*, s'anomenen **segelladors**, o en anglès *sealers*.

Implementació de *Proof of Authority* a *Ethereum*

La plataforma d'*Ethereum* no fou pensada per a implementar *Proof of Authority*, doncs es va pensar per ésser una xarxa oberta i no una permisionada. Tot i així, varis desenvolupadors van pensar com fer funcionar el protocol en xarxes permisionades usant *Proof of Authority*.

Desenvolupadors del client de *Ethereum* en C++ Parity van desenvolupar el protocol de *Proof of Authority* anomenat Aura, mentre que els desenvolupadors del client d'*Ethereum* en Go Go Ethereum o `geth` van desenvolupar el protocol de *Proof of Authority* anomenat Clique.

Després d'avaluar ambdos implementacions de mecanismes de consens del tipus *Proof of Authority* a *Ethereum*, hem escollit Clique per davant de Aura per motius d'eficiència i simplicitat a l'hora de la seva implementació.

Nota: La nostra elecció també ve donada pel resultat d'una investigació que justament comparava l'elecció d'aquests mètdoes des del punt de vista de sistemes distribuïts

---

Clique com a *Proof of Authority*

La implementació de Clique permet de forma senzilla evitar també abusos d'autoritat per part dels segelladors o *sealers*.

La implementació funciona de la següent manera:

S'escull una llista de segelladors al principi de la *blockchain*. Cada segellador s'identifica per la seva adreça d'*Ethereum* derivada a partir de las seva clau privada. S'ordena la llista de forma lexicogràfica.

El primer segellador de la llista serà l'encarregat de minar en primer lloc, el segon en segon lloc, i així per tants segelladors com hi hagi.

En cas que algun segellador no hi sigui a la xarxa quan sigui el seu torn, el següent segellador prendrà el seu torn després d'un temps d'espera. Si aquest tampoc hi és, el torn serà cedit al següent i així consecutivament.

En el cas de que dos segelladors votin alhora, només s'agafarà el bloc d'aquell segellador més proper al segellador del torn actual.

Per evitar abusos d'autoritat, un segellador només podrà segellar N/2 blocs seguits. On N és el nombre de segelladors.

Nota: Clique no sols l'implementa el client Go Ethereum sinó també Pantheon (implementació en Java d'un client de *Ethereum* amb finalitats d'usar-se en xarxes permisionades)

Pantheon està implementat per PegaSys, un equip de ConsenSys dedicat a blockchains empresarials.

Periode de blocs

El periode de blocs determina cada quan s'afegirà un nou bloc a la *blockchain*.

A la xarxa de *Ethereum*, amb el sistema de *Proof of Work*, el mecanisme està configurat per a que la dificultat d'aquest faci que surtin blocs amb un període de 15 segons. És a dir, un bloc nou cada (aproximadament) 15 segons.

A la nostra xarxa, proposem també seguir amb un període de 15 segons per bloc, afegint una peculiaritat. Donat que la xarxa és possible no sigui activa lo suficient com per afegir un nou bloc de transaccions cada 15 segons, volem evitar blocs buits.

Per això, proposem que els segelladors només segellin quan detectin transaccions a la xarxa que han d'ésser validades. D'aquesta manera si en un espai de temps no hi ha activitat a la xarxa, els segelladors no afegiran nous blocs a la *blockchain*, reduint així la mida d'aquesta.

Nota: Un bloc buit ocupa, en format descomprimit, 606 bytes.

Identificador de la xarxa

Per tal de poder sincronitzar la xarxa de *Ethereum* adient, cada xarxa de *Ethereum* ha de definir un identificador de xarxa. Aquest és un nombre únic per cada xarxa que permet que quan sol·licitem informació (blocs, transaccions, ...) a altres nodes, aquests ens retornin la informació de la xarxa que estem interessats.

La xarxa principal de *Ethereum* usa l'identificador `1`, i els identificadors `2` al `4` s'usen per a les xarxes de prova d'*Ethereum* de *Morden*, *Ropsten* i *Rinkeby* respectivament.

Nota: Podeu consultar una llista no exhaustiva de identificadors de xarxes de *Ethereum* públiques conegudes:

https://ethereum.stackexchange.com/q/17051

Noms i identificadors de xarxa a Block Valley

Hem decidit seria bona idea agafar els cims més alts de cada parròquia d'Andorra i usar-los tant com a nom de xarxa, com per el seu identificador.

El nom de la xarxa serà aleshores el nom del pic, i l'identificador serà l'alçada del pic, en metres per sobre del nivell del mar.

Nota: L'alçada serà obtinguda de fonts oficials com cartografia.ad i s'arrodonirà a l'enter més proper en cas que l'alçada contingui decimals.

Així doncs, començarem agafant aquella parròquia amb el cim més baix de totes les parròquies. En el cas que desitgem llançar una nova xarxa amb diferents paràmetres o regles de consens que no siguin compatibles amb la xarxa anterior, agafarem el pic següent i així consecutivament.

Muntanya Nom de la xarxa	**Alçada (msnm)** Identificador de xarxa	**Parròquia**
enclar	2383	Andorra la Vella
monturull	2754	Sant Julià de Lòria
pessons	2850	Encamp
cabaneta	2863	Canillo
portella	2905	Escaldes-Engordany
estanyo	2915	Ordino
comapedrosa	2946	La Massana

Important: La xarxa actual és **enclar**, amb identificador de xarxa **2383**

Portes d'accés a la xarxa

Quan un nou node vol connectar-se a una xarxa P2P, necessita conèixer un altre node per a entrar-hi.

Per automatitzar aquest pas, les xarxes P2P usualment defineixen portes d'entrada, que són servidors que coneixen a alguns nodes de la xarxa P2P i redirigeixen els nodes nouvinguts als nodes que ells coneixen de manera que els nodes nouvinguts puguin connectar-se sense haver de demanar per vies externes l'adreça d'un node existent.

Important: En *Ethereum*, les portes d'accés a la xarxa es denominen **bootnodes**

S'identifiquen amb una adreça IP, un port i una clau pública ECDSA (aquesta última a efectes de validar la seva identitat).

El format és: `enode://pubkey@ip:port`. Per exemple:

`enode://a979fb57[...]3bec910127f134779fbcb0cb6d3331163c@52.16.188.185:30303`

Portes d'accés a les xarxes principals de *Ethereum*

A la xarxa principal de *Ethereum*, les portes d'accés a la xarxa venen *hard coded* al codi dels clients de *Ethereum*. D'aquesta manera, descarregant l'aplicació, podem accedir a les portes d'accés a la xarxa sense fer cap petició addicional. D'aquesta manera ens assegurem que si hem verificat correctament l'integritat de l'aplicació descarregada, també hem verificat l'integritat alhora de les portes d'entrada a la xarxa.

Nota: En el cas de la implementació de *Ethereum* de Go Ethereum, les portes d'entrada de la xarxa principal és troben definides en les següents línies:

https://github.com/ethereum/go-ethereum/blob/v1.8.20/params/bootnodes.go#L19-L31

Portes d'accés a la xarxa de Block Valley

Per poder usar un client de *Ethereum* sense modificacions, no inclourem les portes d'entrada o *bootnodes hard-coded* dins els clients.

Per a especificar les portes d'entrada, aleshores, desarem aquests *bootnodes* al fitxer de configuració de l'aplicació. També es poden especificar a través de la línia de comandes

Nota: A les imatges de *Docker* de **Block Valley**, ja s'ha copiat la configuració adient per tal que no calgui especificar els *bootnodes* manualment

Important: Actualment, a la xarxa **enclar** ja hi tenim dos *bootnodes*, un a Andorra, a les oficines d'una de les empreses del consorci i l'altra a Espanya

Moneda de la xarxa

A *Ethereum*, existeix una moneda nativa, els *ethers*, que s'usen com a moneda per transmetre valor i com a mètode de pagament de les comisions de cada transacció de la xarxa.

Denominació

A **Block Valley**, també tindrem aquesta moneda nativa, que pasarem a anomenar **Block Valleys Andorrans** (BVA)

Nota: Les subunitats s'anomenaran igual que les de *Ethereum*

http://ethdocs.org/en/latest/ether.html

Emissió de moneda

Donat el mecanisme de consens de *Proof of Authority*, no cal incentivar els segelladors amb nova moneda. Els segelladors voluntariament participen en la xarxa. Per tant, cada bloc que es generi no generarà noves unitats de moneda, a diferència de la xarxa principal de *Ethereum*

Important: L'emissió de moneda, doncs, és fixada, un cop, i a l'inici de la xarxa.

Al primer bloc, l'anomenat bloc gènesis o *genesis block* es precarreguen un donat nombre de comptes amb un donat nombre de BVAs.

Ús de la moneda

La moneda, igual que els *ethers* a tota xarxa *Ethereum*, serveix no només per transferir valor si aquesta en té, sinó també per pagar les comisions de les transaccions emeses a la xarxa.

En el cas de **Block Valley**, la moneda ens servirà per controlar l'ús que s'en fa de la *blockchain*:

- Tal i com s'ha descrit a l'apartat anterior, a l'inici de la *blockchain* es precarreguen uns determinats comptes amb un determinat nombre de *BVAs*
- En el moment que algú vulgui participar a la xarxa, haurà de sol·licitar *BVAs* per poder moure aquests *BVAs* a un altre usuari o per crear contractes intel·ligents o *smart contracts*.
- Sense *BVAs* ningú pot participar activament a la xarxa, doncs no podrà pagar les comissions de les transaccions que cal fer per participar-hi.

Per tant, a través de la gestió dels comptes inicials de *BVAs*, podrem controlar qui són de la xarxa. Per exemple, podrem requerir que abans d'obtenir *BVAs* calgui demostrar l'identitat de l'usuari.

D'aquesta manera evitem fraus alhora d'obtenir *BVAs* i en podem contrar el seu ús indegut.

Nota: Ara per ara, els *BVAs* de la primera xarxa de proves, **enclar**, estàn en un compte a expenses de decidir la política d'ús de la xarxa.

2.1.2 Funcionament de la xarxa

Suggerència: Abans de llegir com funciona la xarxa, es recomana haver llegit el capítol de *paràmetres de la xarxa*, sobretot si no es té experiència en xarxes permisionades

---

Tipus de nodes

La xarxa es forma principalment de nodes d'*Ethereum*. Donat que hem escollit *Proof of Authority* com a mecanisme de consens, tindrem dos tipus de nodes: els nodes regulars i els nodes segelladors.

A més, també tindrem els nodes anomenats *bootnodes*, que serviran com a porta d'entrada per a nous nodes.

A continuació explicarem la funcionalitat de cadascun dels nodes.

Nodes

Els nodes de la xarxa s'encarreguen de connectar amb altres nodes per comunicar missatges (blocs i transaccions) i desar una còpia local de la *blockchain*.

Com hem comentat a l'apartat de *Consens*, el fet de triar el mecanisme de consens de *Clique com a Proof of Authority*, fa que usem en concret el client de *Ethereum* `geth` o Go Ethereum per als nodes de la xarxa.

Important: Usarem Go Ethereum (`geth`) com a implementació de client de *Ethereum* de la nostra xarxa.

Dintre dels nodes de la xarxa, trobem principalment dos tipus de nodes segons la seva funcionalitat. Els dos tipus de nodes executen el mateix codi, pero s'usen per a diferents funcions.

Nodes regulars

Sincronitzen i validen la *blockchain* localment i participen en la retransmissió d'aquesta per la xarxa.

S'usen com a punt d'accés a la *blockchain* pels usuaris que la vulguin usar.

Important: Els **nodes regulars** repliquen la *blockchain*, la validen de forma independent i permeten als usuaris accedir a les dades d'aquesta de forma local a través de la còpia local descarregada.

Nodes segelladors o *sealers*

Realitzen les mateixes funcions que els nodes regulars, però a més, tenen un compte configurat per a emetre blocs.

Aquest compte ha d'ésser un dels comptes autoritzats per a emetre blocs.

Important: Els **nodes segelladors** o **sealer nodes** realitzen les mateixes funcions que els **nodes regulars**, però a més, s'encarreguen d'afegir nous blocs a la *blockchain* realitzant signatures digitals un cop configurat el compte de segellador.

Nota: Els comptes autoritzats per a emetre blocs es defineixen al bloc de gènesis però aquests poden ésser modificats a posteriori realitzant votacions per afegir o eliminar comptes autoritzats.

Bootnodes

Els *bootnodes* o nodes d'inici, són nodes reduïts que només tenen funcions de xarxa i no emmagatzemen cap *blockchain*.

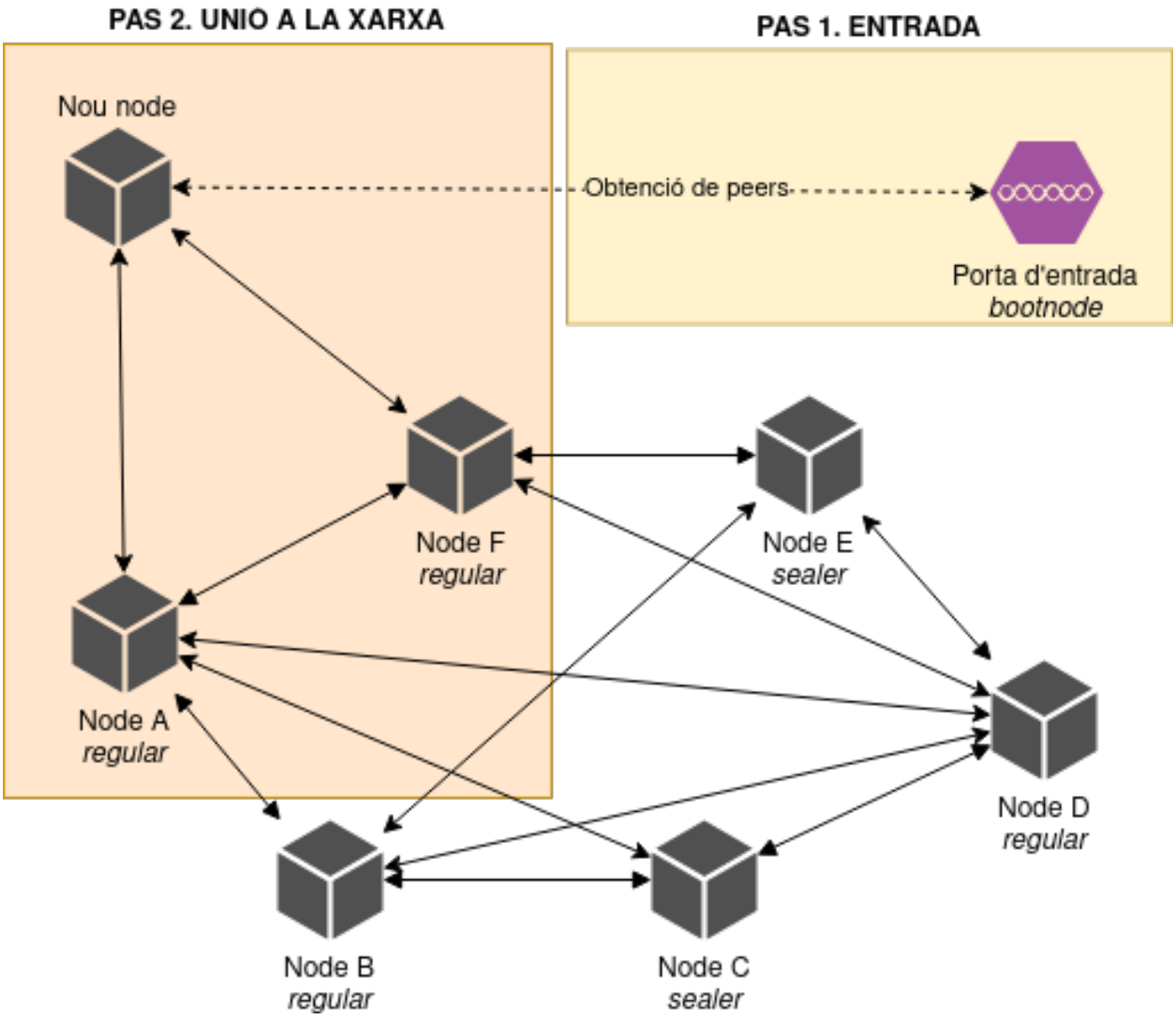
El que proporcionen és una porta d'entrada a la xarxa per a nous nodes, indicant-lis on són els actuals participants de la xarxa.

Avís: Per motius de seguretat, els *bootnodes* requereixen d'un parell de claus ECDSA donat que són la porta d'entrada a la xarxa.

Important: Els **bootnodes** no tenen desada la *blockchain*, només fan funció de porta d'entrada. Connecten als nous nodes amb els nodes que ja existeixen de la xarxa.

Un node normal, també fa de **bootnode**.

2.1.3 Visió global del funcionament



Accés a la xarxa

El primer que succeeïx quan un nou node (sigui regular o segellador), es vol connectar a la xarxa és cercar on connectar-s'hi. Al éssser una xarxa P2P, no hi ha un servidor central, hem de cercar els altres nodes o *peers*.

Per a fer-ho, demana als *bootnodes* existents, que seria la única figura central disponible coneguda, quins participants hi ha en aquesta per a connectar-se a ells.

Al fer-ho, els *bootnodes* s'apunten que aquest nou node acaba de formar part de la xarxa, per indicar als futurs nodes que vinguin l'adreça d'aquest junt amb la dels anteriors.

Sincronització de la xarxa

Un cop el node s'ha unit a la xarxa, comença a demanar blocs des de l'últim punt on s'hi va quedar. Si és un nou node, aquest punt serà el bloc 0, el bloc gènesis, on es defineix el començament de tota *blockchain*.

No només descarregarà una còpia de la *blockchain*, sinó que també comprovarà que els blocs s'hagin signat pels segelladors autoritzats, i les transaccions siguin vàlides. D'aquesta manera permetem la comprovació automàtica i independent de les dades de la *blockchain*

Ús de la xarxa

Un cop tenim el node sincronitzat a l'últim bloc disponible, ja podem fer-ne ús. Per a fer-ho, haurem d'habilitar la interfície JSON/RPC per a fer crides amb el protocol RPC estàndard de *Ethereum* cap al node.

Nota: Les imatges de *Docker* dels nodes ja venen configurades per habilitar la interfície RPC per defecte. Només s'ha d'exposar el port 8545.

2.1.4 Ús de la xarxa

Per a usar la xarxa, podem fer-ho de la mateixa forma que a la xarxa principal, tenint en compte que ens hem de connectar a un node de la xarxa de **Block Valley** i que canviarà l'identificador de xarxa.

Aqui trobareu tota la informació per poder tenir un node de la xarxa i per poder usar-la.

Monta el teu node (amb Docker)

Per reduir la complexitat de tenir un node, hem preparat imatges de Docker el més configurades possible per a que muntar el teu propi node sigui tan fàcil com executar una comanda.

El registre d'imatges de Docker

Tenim totes les imatges de Docker publicades a Dockerhub sota l'usuari de l'organització `blockvalley`

https://hub.docker.com/u/blockvalley

Les etiquetes de les imatges

Cada imatge té tota la configuració possible per executar amb el mínim esforç posible cada tipus de node.

Això implica que algunes imatges tenen informació de la xarxa a la qual ens hi volem connectar. És per això que l'etiqueta d'una imatge que requereixi configuració de xarxa serà el nom de la xarxa on ens hi volguem connectar.

Les imatges

Actualment, disposem de les següents imatges de Docker:

Node

Un node estàndard, amb funcions típiques d'un node de *blockchain*. També té la interfície JSON-RPC habilitada per a poder usar-lo com a punt d'accés local a la xarxa.

Nota: Aquesta imatge conté informació de xarxa, de forma que l'etiqueta que s'usi per seleccionar la imatge determinarà a quina xarxa és connecta el node.

És per aquest motiu que no existeix l'etiqueta 'latest'

A les comandes següent ens connectarem a la única xarxa disponible, la xarxa **enclar**.

Suggerència: Les imatges dels nodes estan basades en la última versió estable del client de *Ethereum* Go Ethereum, que alhora usen la base de Alpine Linux

https://hub.docker.com/r/ethereum/client-go/

A data d'avui (10 d'oct., 2019), és el client de Go Ethereum versió 1.8.20

Funcions

- Manté una còpia de la *blockchain*
- Verifica que totes les dades de la *blockchain* siguin correctes
- Transmet totes les dades als *peers* de la xarxa P2P als quals roman connectat
- Permet consultar localment dades de la *blockchain*
- Permet interactuar amb la xarxa *blockchain* (per exemple, enviar transaccions)

Execució

Per executar la imatge del node, podem usar la següent comanda de Docker::

```
docker run -p 30303:30303 -p 8545:8545 blockvalley/enclar-node
```

El port 30303 s'utilitza per a les comunicacions de la xarxa P2P, mentre que el port 8545 l'usem per connectar-nos amb l'API JSON-RPC i interactuar amb la xarxa (per exemple amb *MetaMask*)

Atenció: Cal exposar el **port 30303** a *Docker* per poder connectar-se a la xarxa i a més, tenir el port `30303` accesible externament des d'Internet (*tenir el port 30303 obert*).

En cas que no tinguem el port obert, no podrem sincronitzar blocs i si és un node nou, només tindrem accesible el bloc gènesis.

Això implica **modificar les regles de ports (NAT) del router usat per a la connexió a Internet**.

Nota: Hem de tenir en compte que si no afegim persistència, un cop eliminem el *container* de Docker, haurem de tornar a sincronitzar. Podem afegir persistència usant els volums de Docker:

```
docker run -p 30303:30303 -p 8545:8545 \
   --mount type=volume,source=node-enclar,destination=/root/.ethereum
   blockvalley/enclar-node
```

Ara si aturem i esborrem el *container* (per exemple, per usar una imatge més actualitzada), podem llançar un nou *container* amb la *blockchain* ja sincronitzada usant les dades del volum.

Paràmetres

Podem afegir paràmetres addicionals de Go Ethereum (`geth`) a l'execució del node, només cal afegir-los darrere de la imatge. Per exemple, per mostrar l'ajuda on es mostren tots els paràmetres disponibles::

```
docker run blockvalley/enclar-node -h
```

Suggerència: En cas de no especificar cap paràmetre, s'usaran els recomenats per defecte, que venen inclosos a la imatge. Aquests inclouen, entre d'altres els detalls per connectar-se a la xarxa (*bootnodes*, id de xarxa, . . .)

Node segellador o *sealer*

El node segellador és una extensió del node regular definit al punt anterior. Aquest node està preparat per a segellar blocs, amb una configuració addicional que només activa el procés de segellat quan hi ha activitat a la xarxa per reduir al màxim els blocs buits.

Tot l'anterior explicat al node aplica igual per al node segellador, excepte les diferències a continuació.

Funcions

Totes les funcions de un node, tal i com s'ha vist al punt anterior més:

- Segellat: emissió de blocs nous realitzant signatures digitals quan hi pertoqui (hi ha activitat i és el torn oportú)

Execució

A diferència del node, cal configurar un compte de *Ethereum* al segellador, que usarem per a signar els nous blocs emesos per nosaltres.

Compte de *Ethereum*

Claus ECDSA

Per a tenir un compte de *Ethereum*, necessitarem un parell de claus ECDSA.

Per a generar un parell de claus, només necessitarem una clau privada ECDSA. Amb la clau privada després en podem generar la pública i d'aquí l'adreça de *Ethereum*.

Atenció: Haurem de guardar aquesta clau privada de la forma més segura possible, doncs algú amb accés a aquesta podria suplantar la identitat de segellador.

Això implicaria problemes de **denegació de servei** en cas que més dels *n/2* segelladors siguin compromesos.

Generar una clau privada es pot fer de forma fàcil amb un *script* de *shell* present al repositori de codi principal

```
./scripts/generate_bytes.sh > private_key.txt
```

L'*script* generarà una clau privada ECDSA de 32 bytes (representada en forma hexadecimal) usant aleatorietat del sistema operatiu (per defecte `/dev/urandom`).

Truc: També es poden usar *wallets* jeràrquics deterministes (HDWallets) a partir d'una llavor o *seed* o bé amb una frase secreta (*mnemonics sentence*) usant el BIP 39 per obtenir un arbre de claus privades.

Per a obtenir aquesta frase, podem usar un generador o inclús usar un dau físic per aconseguir la màxima aleatorietat possible

---

Contrasenya de protecció

Finalment, haurem de definir una contrasenya per bloquejar aquest compte. D'aquesta manera no tindrem la clau privada desada en clar.

Aquesta també la podem generar de forma fàcil amb un *script* de shell::

```
./scripts/generate_password.sh > password.txt
```

Amb la clau privada i contrasenya per protegir-la, ja podem preparar el compte a usar per segellar.

Configurar el compte del segellador

Per això, ens cal **usar persistència**, per a que si aturem o esborrem el *container*, seguim tenint el compte configurat per a emetre blocs.

Important: D'altra manera, si el *container* s'atura i s'elimina, haurem de sincronitzar la *blockchain* de nou i configurar el compte per a signar els blocs de nou.

Això pot implicar aturades de la xarxa si succeeix a un nombre de segelladors superior a *n/2* on *n* és el nombre de segelladors.

Tècnicament no és necessari, pero a nivell operatiu **és imprescindible**.

Usarem volums de docker per a persistir dades del *container* que executem.

Per a configurar el *volum*, que anomenarem `sealer-enclar`, fem servir un script de *shell* disponible al repositori de codi principal de **Block Valley**

Executem la següent comanda:

```
./scripts/init_account.sh sealer-enclar private_key.txt password.txt \
> address.txt
```

Aquest script executarà la comanda d'importació de compte de `geth` i desarà el resultat al volum seleccionat. Com a resultat, si no hi ha cap error, retorna l'adreça de *Ethereum* del compte importat.

Podem veure aquesta mostrant el fitxer `address.txt` que acabem de generar:

```
cat address.txt
0x5ea1e4d01187347ff9b3bc672f4a628c9c75d007
```

Important: Anoteu aquesta adreça, l'haureu d'enviar al consorci per a poder ésser un node segellador del consorci un cop la majoria hi estigui d'acord.

Ara per ara, el procés de convertir-se en segellador és manual. Envieu un correu a amb l'assumpte `Alta sealer`.

En el futur, el procés serà automàtic (*Segelladors*)

Ja hem inicialitzat el volum `sealer-enclar` amb la clau privada del fitxer `private_key.txt` protegida amb la contrasenya especificada a `password.txt`.

Ara ja podem llançar el nostre node segellador::

```
docker run -p 30303:30303 \
    --mount type=volume,source=sealer-enclar,destination=/root/.ethereum \
    --env PASSWORD=$(cat password.txt) \
    blockvalley/enclar-sealer
```

Un cop sincronitzat, el nostre node ja estarà llest per a emetre nous blocs. Haurem d'esperar, això sí a que sigui assignat com a segellador per majoria de vots dels segelladors actuals.

Neteja de fitxers

Perquè és important

Hem d'esborrar el fitxer `private_key.txt` amb la clau privada per evitar la fuga d'aquesta.

El fitxer de `password.txt` no s'ha d'esborrar doncs ens permet usar la clau privada al arrencar el node sense haver de manualment introduïr la contrasenya de xifrat de la clau privada.

Sobretot cal copiar en un lloc segur aquesta informació abans d'esborrar-se, doncs si es perd serà irrecuperable

Per seguretat, copiarem els fitxers amb dades sensibles **a través d'un mètode segur, com físicament a un dispositiu o via SSH o altres mètodes xifrats** els fitxers `private_key.txt` i `password.txt`.

```
cp password.txt private_key.txt /media/myusb
scp password.txt private_key.txt me@myserver:~/safeplace/
```

Nota: Reemplaceu `/media/myusb` o `me@myserver:~/safeplace/` per una destinació segura per a desar els fitxers de la vostra elecció

Necessitem conservar el fitxer *password.txt* per a quan s'hagi de reiniciar el *container* de Docker pugui desbloquejar el compte, però podem eliminar de forma segura el fitxer `private_key.txt` (un cop l'hem copiat)

```
shred -n 10 private_key.txt
```

En resum

Comandes a executar:

```
# Creació de compte
./scripts/generate_bytes.sh > private_key.txt
./scripts/generate_password.sh > password.txt
# Inicialització de volum de Docker amb compte
./scripts/init_account.sh sealer-enclar private_key.txt password.txt
# Execució
docker run -p 30303:30303 \
   --env PASSWORD=$(cat password.txt) \
   --mount type=volume,source=sealer-enclar,destination=/root/.ethereum \
   blockvalley/enclar-sealer
# Deseu la clau privada private_key.txt en un lloc segur
# cp private_key.txt password.txt /media/myusb
# cp private_key.txt password.txt me@myserver:~/safeplace
# La contrasenya password.txt també!
# Eliminem el rastre per seguretat
shred -n 10 private_key.txt
```

Usant MetaMask

Podem usar l'extensió de navegador MetaMask per a usar la nostra xarxa en el nostre navegador d'Internet preferit. Està disponible per a *Google Chrome*, *Firefox* i *Opera*

Un cop instal·lada l'extensió, hem de configurar-lo per a què es connecti a la xarxa de **Block Valley**, ja que per defecte es connectarà a la xarxa principal de *Ethereum*.

Configuració de MetaMask

Per a obrir la configuració de xarxa, obrim la extensió de **MetaMask** fent clic sobre la seva icona i **desbloquegem el nostre compte**.

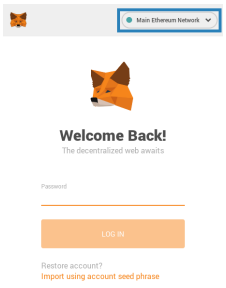


Fig. 1: Pantalla de desbloqueig de MetaMask

Nota: En el cas que no tinguem compte, ens asistirà per crear-ne un. Ens hem de desar les (12 o 24) paraules clau del nostre nou compte, doncs seràn les que ens permetran accedir-ne a aquest. En cas que les perdem, **no hi haurà**

manera possible de recuperar el compte.

Un cop tenim accés a la pantalla principal, fem clic sobre el seleccionable de xarxes per poder afegir-ne una de nova, la de **Block Valley**.

Suggerència: El seleccionable de xarxes el podeu desplegar fent clic sobre la secció marcada en blau a la *figura anterior*.

Al desplegable, seleccionem la opció **Custom RPC**

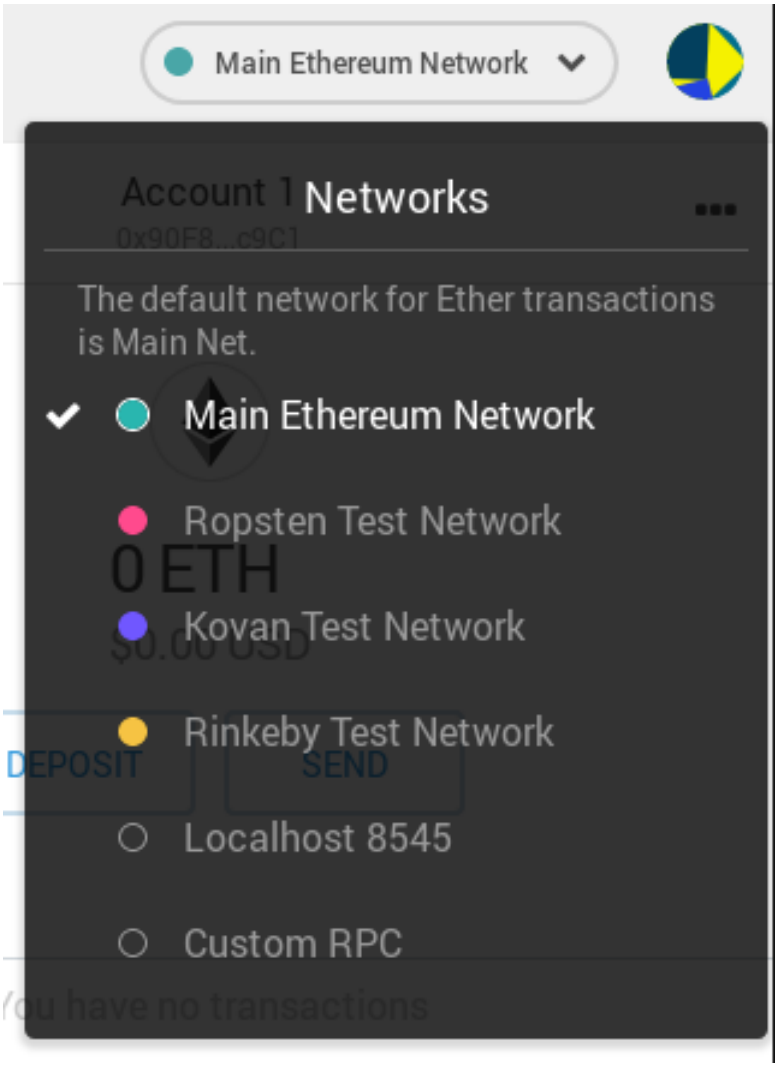


Fig. 2: Desplegable de selecció de xarxes de MetaMask

Important: Si no hem **desbloquejat el nostre compte** (introduït la contrasenya), no ens apareixerà la següent pantalla.

Ens desplacem cap a la opció de *New Network*. Allà, introduïm les dades del node on es conectarà el MetaMask per accedir a la xarxa. Només cal emplenar de forma obligatòria la URL del node en format següent::

```
http://<host>:<port>
```

On *host* és el host amb el node de la xarxa executant-se i *port*, el port de la interfície JSON-RPC (per defecte, 8545).

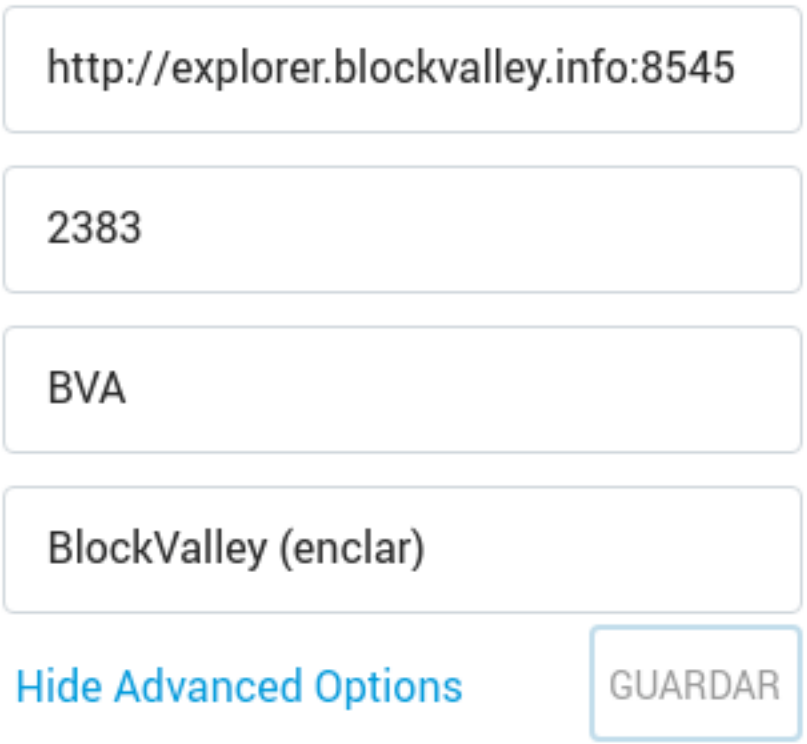


Fig. 3: Configuració de la nova xarxa de **Block Valley** a MetaMask

Avís: Donat que el client que usem per a la xarxa, `geth`, no implementa capa de HTTPs, les consultes cap al node no viatjaran xifrades.

Això pot ésser un risc de seguretat si el node està exposat a Internet. De tota manera, donat que les dades que es consultaràn són públiques a la *blockchain* de **Block Valley**, no suposa un greu risc de privacitat o seguretat, si bé la millor opció és tenir localment un node propi.

Nota: Si no tenim cap node, podem usar-ne un públic, com el que apareix a la figura. En el cas que en disposem d'un, introduïm les dades d'accés (bàsicament només cal *host* i *port*).

Finalment, només hem de seleccionar la xarxa **Block Valley** al desplegable de xarxes que hem fet servir anteriorment per poder realitzar transaccions en aquesta xarxa.

Farem servir el node que hem configurat per enviar i rebre informació de la *blockchain* subjacent.

Nota: A les imatges s'ha especificat l'identificador de la xarxa *enclar* i un node públic d'aquesta. La mateixa configuració s'aplica a altres xarxes, usant un node connectat a una xarxa diferent i un identificador de xarxa diferent

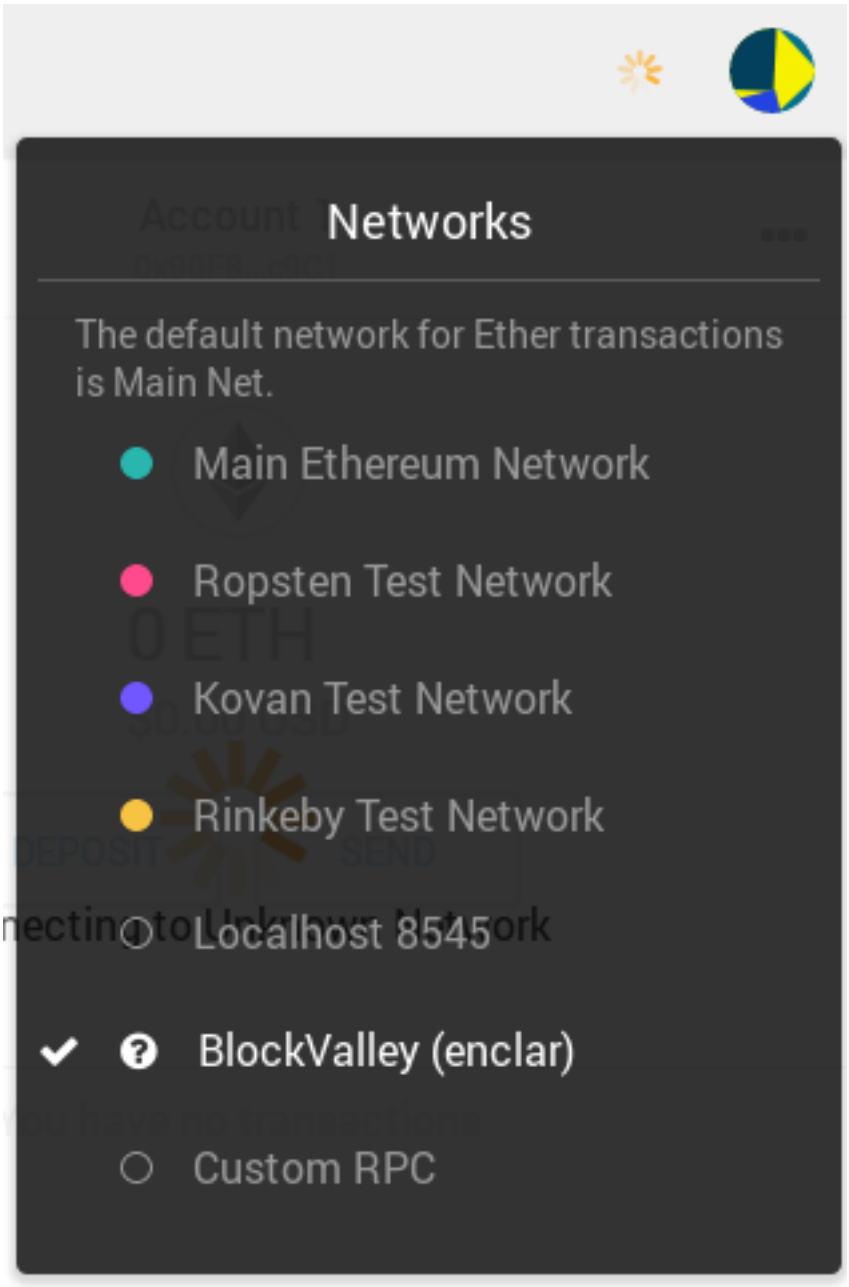


Fig. 4: Selecció de la xarxa de **Block Valley** configurada.

(o omentent-lo si el node només és connectat a una sola xarxa)

2.1.5 Governança

Donat que es vol tenir una xarxa controlada, hem d'establir les regles de governança. A banda dels paràmetres especificats al document de *Paràmetres de la xarxa*, també haurem de definir altres aspectes, com la distribució de fitxes o *tokens*, els requisits per a ésser un node *segellador*, entre d'altres.

Nota: Tot el descrit a continuació està subjecte a canvis un cop tots els membres de la xarxa hi estem d'acord.

Per a contactar amb tothom, podeu fer-ho a través del correu

Distribució del *software*

Les xarxes P2P estan formades per nodes, que no són res més que computadors executant una aplicació / **software específic**. És important per a assegurar la estabilitat de la xarxa que aquest *software* no contingui cap vulnerabilitat o *bug* que suposi un perill a la xarxa.

Nota: Aquest és un dels motius pels quals hem decidit usar el client de Go Ethereum (*entre d'altres*). Aquest és un client d'*Ethereum* que s'usa a la xarxa principal en la gran majoria dels seus nodes

Té una àmplia comunitat de desenvolupadors al projecte *open-source* que fan que el codi sigui auditat a diari, incrementat la seguretat del *software*.

Per això, evitarem també aplicar canvis sobre el codi de Go Ethereum, per a poder mantenir els nodes al xarxa al dia de les actualitzacions de seguretat del *software* sense haver d'invertir temps en aplicar canvis específics de la nostra xarxa sobre el client de la xarxa principal.

Ara per ara, el *software* usat per tenir un node a la xarxa, sigui del tipus que sigui (*bootnode*, *sealer* o segellador o un node regular) és exactament la última versió estable de Go Ethereum. Això sí, **amb una configuració específica** per a la nostra xarxa (l'especificada a *Paràmetres de la xarxa*).

Aquesta configuració bàsicament comprèn dues peces principals:

- Els **bootnodes** o portes d'entrada a la xarxa.
- El **bloc gènesis** amb les regles de la xarxa. El primer bloc de la cadena.

Per assegurar que tothom és a la mateixa xarxa, ens haurem de posar d'acord tots en quins són aquests dos punts. Sobretot, en quin és el **bloc gènesis** doncs sinó, ens trobarem en diferents cadenes i no podrem tenir una xarxa uniforme on tothom compartim la mateixa informació.

Estat actual

El braç tècnic de BTC Assessors (BTCA Labs) publica aquestes configuracions per a la xarxa de proves **enclar** al repositori de la organització principal

https://gitlab.com/blockvalley/blockvalley

Nota: Tenir una còpia del repositori a altres servidors de *Git* seria una bona praxis per evitar que nous membres no puguin accedir a la xarxa si es produeix una caiguda al servei de *Git* de *GitLab*.

Tenim un mirall del repositori a *GitHub*. https://github.com/blockvalley/blockvalley

La última configuració disponible sempre es trobarà a la branca **master**.

Important: Els *commits* hauràn d'estar signats digitalment per un dels dos desenvolupadors següents:

- D214 E5C6 6C18 EEC5 8B21 11CE 1441 FA43 5DE6 AC64

  David LJ <>

- CB79 2CB0 153C 4121 72B6 144C 37FA 2688 86E3 BB13

  Carlos GC <>

Així mateix, aquesta documentació on es fa referència a aquestes claus també està signada digitalment (els *commits*) al seu repositori de *GitLab* (https://gitlab.com/blockvalley/docs) per una de les claus anteriors.

Per a fer-ho més fàcil, es publiquen també imatges de *Docker* al registre oficial amb aquestes configuracions incloses. Per a més informació sobre el seu ús, consulteu el document de *Monta el teu node (amb Docker)*.

Avís: Les imatges de *Docker* encara no estàn signades digitalment per una de les claus anteriors (ni per cap altra).

Portes d'entrada a la xarxa

Un punt crític de la xarxa és la porta d'entrada. Quan un nou node vol formar part de la xarxa, ha de consultar a un o més nodes predefinits on són la resta de nodes. Sino, cadascú hauria de preguntar a algun membre de la xarxa quin és el seu node per entrar-hi a formar-ne part, afegint fricció al procés d'entrar a la xarxa.

Estat actual

Actualment, les dues portes d'entrada, una a Andorra la Vella i l'altre a França, es troben gestionades per BTCA Labs i configurades per defecte a les imatges de *Docker* distribuïdes.

Segelladors

Els nodes *segelladors* s'encarreguen de decidir en cas d'empat, quina és la transacció que entrarà a un bloc si s'emeten dos transaccions del mateix origen amb diferents destinacions (*double spend*).

Truc: Per a més informació, consulteu el document de *Consens* i tipus de nodes *Funcionament de la xarxa*

Estat actual

Hi ha dos segelladors, un a Andorra i l'altre a França, ambdós gestionats per BTCA Labs.

Les adreces d'aquests són (respectivament):

- `0x5ea1e4d01187347ff9b3bc672f4a628c9c75d007` (`btca-sealer01`)
- `0xd9c911ac1e861e9a3db2fbfd77b13911519368ab` (`btca-sealer02`)

Nota: Com veieu (la primera) l'hem buscada per a que comenci per la paraula *sealer* (`5ea1e4`) en llenguatge l33t

Estat futur

Cada node de la xarxa podrà auto esdevenir *segellador* un cop hagi demostrat el seu compromís amb aquesta. Això significarà, en poques paraules, romandre connectat a la xarxa un gran percentatge del temps (proper a 24/7) i no obstaculitzar-ne el seu funcionament (no participar-ne, emetre blocs invàlids, . . .).

Distribució de *tokens*

Per motius d'implementació, els *tokens* de la xarxa s'emeten a priori (al **bloc gènesis**) i no es poden tornar a generar.

Estat actual

El membre impulsor del consorci BTCA Labs té en possessió tots els *tokens* de la xarxa de proves **enclar** a esperes d'aplicar el procediment futur un cop decidit pels membres del consorci.

Nota: A més, els nodes segelladors, quan emetin un nou bloc, rebran com a recompensa les comisions de les transaccions contigudes als blocs que emetin.

La comisió d'una transacció és el gas comprat per poder executar la transacció.

Això recompensa aquest tipus de nodes per la tasca de mantenir un node operatiu a la xarxa amb alta disponibilitat.

Estat futur

Els *tokens* es distribueixen als usuaris de la xarxa en funció de la seva participació (tenir un node connectat, usar la xarxa de forma correcta, . . .).

Per a assegurar-ne la justa distribució en funció d'aquests paràmetres a definir usarem *smart contracts* a la pròpia *blockchain* on s'hauràn de repartir.

CAPÍTOL 3

Indexs i taules

- genindex
- modindex
- search