UNIVERSITÀ DEGLI STUDI ROMA TRE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi Di Laurea

Analisi, progettazione e prove sperimentali di un FulgurHub in TypeScript

Laureando

Federico Ginosa
Matricola 457026

Relatore
Alberto Paoluzzi

Correlatore
Federico Spini

Anno Accademico 2017-2018

Ad Ada Lovelace

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

2008 whitepaper Bitcoin Nel 2008 viene pubblicato il whitepaper di Bitcoin, una proposta di soluzione al problema della doppia spesa basata sull'utilizzo di una rete peer-to-peer [9].

2009 pubblicazione protocollo Il 2009 vede la nascita della prima cryptovaluta, bitcoin, con la pubblicazione dell'implementazione del protocollo Bitcoin. Gli standard variano, ma sembra essersi formato un consenso nel riferirsi con Bitcoin maiuscolo al protocollo e con bitcoin minuscolo alla moneta in se [11].

Bitcoin 10 anni dopo Negli anni successivi decine di protocolli alternativi a Bitcoin sono fioriti. Le cryptovalute da argomento di nicchia hanno visto una costante crescita di adozione. Dal 2009 fino a oggi il numero di transazioni quotidiane è cresciuto più che linearmente, raggiungendo il suo attuale picco storico nel dicembre del 2017, con più di 450K transazioni in un giorno [12].

Limiti architetturali Questo crescente interesse nei confronti delle cryptovalute si scontra però con i limiti architetturali relativi alla scalabilità della blockchain. Con gli attuali parametri di blocksize e blockinterval (1 megabyte

e 10 minuti), il throughput massimo è compreso tra le 3 e le 7 tps (transazioni per secondo). Portando questi due parametri all'estremo, 1 megabyte e 1 minuto, si riuscirebbe a decuplicare il throughput, senza sacrificare il protocollo in termini di sicurezza [4]. Sebbene 30 - 70 tps rappresenterebbero un fondamentale miglioramento tecnologico di Bitcoin, il throughput raggiunto non sarebbe comunque confrontabile con quello di sistemi centralizzati analoghi come il circuito VISA, con le sue 56K tps [13].

Soluzioni Diverse sono le soluzioni proposte per risolvere i problemi di scalabilità della blockchain e possono essere suddivise in tre categorie:

- **Algoritmo di consenso** Alla base del whitepaper di Satoshi Nakamoto c'è il Proof Of Work. Modificando il meccanismo alla base della ricerca del consenso è possibile migliorare la scalabilità della blockchain. Ad oggi diverse sono le alternative proposte [6], [1], [7].
- **Sharding** Questo concetto non è nuovo nel mondo dei database. L'idea è quella di suddividere la blockchain in più parti. La ricerca del consenso avviene in ciascuno di queste parti. Anche da questo punto di vista diversi sono i lavori e le proposte [8], [14].
- **Off-chain** Layers è un famoso pattern architetturale. Un forte impiego di questo pattern è stato fatto nell'ambito del networking, vedi pila ISO/OSI. La scalabilità off-chain si realizza costruendo un secondo layer sopra alla blockchain, che permetta di ereditare le sue caratteristiche (sicurezza e distribuzione), aggiungendone delle altre, come la scalabilità.

Questi tre diversi approcci alla scalabilità della blockchain non sono in contrasto l'uno con l'altro, ma anzi possono essere applicati assieme in maniera sinergica. Nel lavoro di questa tesi ho approfondito l'ultima categoria, la scalabilità off-chain. In particolare mi sono occupato delle seguenti attività:

- Analisi dello stato dell'arte relativa a soluzioni di scalabilità off-chain
- Realizzazione di un canale di pagamento inestinguibile (IPC)

- Analisi, progettazione e sviluppo di FulgurHub
- Prove sperimentali di FulgurHub

In questa tesi il capitolo due tratta il background necessario, in particolare si approfondisce il design di un payment channel e si introduce l'architettura di FulgurHub. Nel terzo capitolo si effettua l'analisi nel dettaglio di FulgurHub. Nel quarto capitolo si descrivono le fasi di progettazione e sviluppo di FulgurHub. Nel quinto capitolo si mostrano le prove sperimentali relative a quanto è stato implementato e si discutono i risultati in termini di performance e scalabilità.

Capitolo 2

Background

2.1 Blockchain

Il problema La blockchain risolve il problema del double spending in un sistema peer-to-peer completamente decentralizzato [9]. Questo permette di memorizzare in maniera immutabile dei pagamenti in un registro pubblico, avendo la certezza che nessuno possa spendere più volte lo stesso token.

Il caso d'uso Il caso d'uso tipico della blockchain è l'invio e la ricezione di pagamenti. La transazione rappresenta un pagamento. Essa può essere immaginata come un arco che unisce due nodi. Il nodo iniziale rappresenta il pagante, il nodo finale il pagato. Tutte queste transazioni vengono memorizzate su un registro pubblico detto ledger. Sebbene le transazioni siano pubbliche, esse vengono descritte anche come pseudoanonime, in quanto pubblico è solo l'indirizzo di paganti e pagati ma non il loro nominativo.

Cos'è la blockchain La blockchain è una lista concatenata di blocchi. Ciascun blocco contiene: l'hash del precedente blocco, il merkle root relativo alla lista di transazioni associate al blocco corrente e un nonce. In Bitcoin un

nuovo blocco viene aggiunto ogni dieci minuti e il merkle root rappresenta una prova succinta di una lista di transazioni di dimensione minore o uguale a 1 megabyte.

Come funziona la PoW I blocchi vengono aggiunti dai miner. I miner sono dei nodi della rete che si occupano di trovare un nonce cha faccia si che l'hash del blocco corrente abbia un numero di zeri iniziali pari a D. Questo valore D rappresenta la difficoltà corrente di mining della rete. La difficoltà è autoregolata dal protocollo e aumenta o diminuisce a seconda del tempo necessario per minare i precedenti blocchi. Un miner che riesce a presentare un nonce e un blocco valido ottiene in cambio le fee delle singole transazioni e una coinbase.

Cos'è uno smart contract Inviare un pagamento in Bitcoin significa sbloccare uno o più UTXO (Unspent Transaction Output). Sbloccare un UTXO significa presentare una prova crittografica della proprietà di un certo token. La verifica della prova crittografica viene effettuata da tutti i nodi della rete eseguendo un ASFND (automa a stati finiti non deterministico). Il protocollo Bitcoin permette di implementare e deployare sulla rete degli automi anche più complessi. Script è il linguaggio di programmazione stack-based non Turing-completo che permette di descrivere questi automi in Bitcoin. Quando la complessità degli automi aumenta, si parla di smart contract, ovvero di contratti che permettono lo sblocco di fondi previa verifica di un insieme complesso di regole.

Smart contract Turing-completi Sebbene abbia senso parlare di smart contract in Bitcoin, l'uso del termine in questo contesto è stato introdotto solo nel 2014, con la pubblicazione del whitepaper di Ethereum [3]. Ethereum è un protocollo che eredita gran parte delle caratteristiche di Bitcoin e in più introduce la EVM (Ethereum Virtual Machine) la macchina virtuale che esegue gli smart contract. Gli smart contract in Ethereum vengono descritti in Solidity, un linguaggio di programmazione C-like Turing-completo. La

turing completezza permette di descrivere un più ampio spettro di regole.

Scalabilità off-chain Nel Capitolo 1 sono stati introdotti i limiti architetturali della blockchain e le tre categorie di approcci risolutivi. La scalabilità off-chain è una delle tipologie di soluzioni atta a superare i limiti di scalabilità della blockchain. Questo approccio riduce sensibilmente le interazioni necessarie sulla blockchain, spostandole fuori di essa, senza compromettere le proprietà di sicurezza.

2.2 State channel

Gli state channel permettono a due parti di modificare in maniera sicura porzioni della blockchain, dette depositi di stato. Questi depositi di stato sono memorizzati all'interno di indirizzi multisignature. Le parti modificano lo stato dello state channel scambiando messaggi off-chain. Questi messaggi descrivono un aggiornamento dello stato, per esempio la prossima mossa in una partita di tris [15].

2.2.1 Payment channel

Un payment channel è una particolare tipologia di state channel. I messaggi scambiati off-chain rappresentano dei pagamenti, ovvero l'aggiornamento del bilancio delle parti. Instaurare un payment channel richiede una sola operazione on-chain da ciascuna parte. L'operazione on-chain viene eseguita su uno smart contract dedicato al singolo payment channel. Questa unica operazione on-chain abilita un numero illimitato di pagamenti off-chain, nei limiti del balance iniziale delle parti. I messaggi off-chain possono essere scambiati mediante qualunque mezzo, comunemente una connessione http. Un payment channel permette dunque di spostare i problemi di scalabilità

dalla blockchain a un server http, ma la letteratura riguardo a come far scalare quest'ultimo è consolidata.

Architettura Un payment channel permette di effettuare un numero illimitato di transazioni off-chain tra due parti. Ciascuna parte deve mettere a disposizione un server http che permetta l'invio e la ricezione di pagamenti. Una delle due parti deploya lo smart contract associato e apre il canale. In un secondo momento la controparte effettua il join del canale, stabilendone la definitiva apertura. In questa progettazione si è presa come riferimento la blockchain di Ethereum.

Deploy Il deploy è la prima fase di inizializzazione. Alice deploya lo smart contract del relativo canale. L'operazione di deployment è richiesta per ciascun singolo payment channel. Questa fase permette di ottenere l'indirizzo di un contratto, che nelle successive fasi verrà adottato per richiamare le operazioni on-chain che si intende richiamare. In questa fase lo stato del payment channel è detta $INIT$.

Figura 1: Deploy on-chain dello smart contract di un payment channel.

Apertura Alice apre il canale e blocca un quantitativo arbitrario di fondi all'interno dello smart contract. Questi fondi rappresentano il bilancio iniziale di Alice. Si fa notare come la fase di deploy e di apertura possano essere svolte con un'unica operazione on-chain. Oltre a depositare i fondi, Alice con questa operazione porta in catena il suo indirizzo ip e l'indirizzo ethereum di Berto. Terminata la procedura lo stato del canale diventa $OPENED$.

Join In un secondo momento Berto effettua il join del canale di pagamento aperto da Alice. Anche questa operazione viene effettuata on-chain. Berto

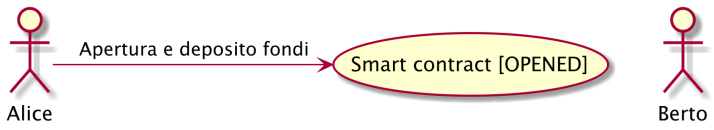

Figura 2: Apertura e deposito fondi on-chain in un payment channel.

deposita i fondi che corrisponderanno al suo bilancio iniziale e porta in catena il proprio indirizzo ip. Con questa operazione il canale è definitivamente stabilito e lo stato passa da $OPENED$ a $ESTABLISHED$.

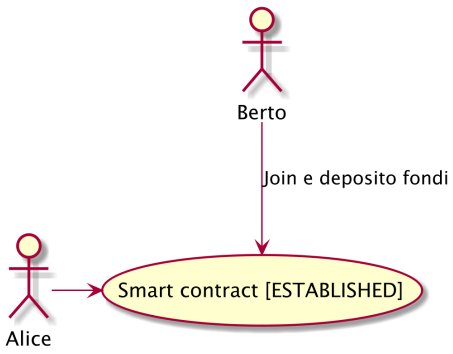

Figura 3: Join e deposito fondi on-chain in un payment channel.

Schema propose/accept I pagamenti off-chain avvengono mediante lo schema propose/accept. Alice (o Berto) propone un aggiornamento dello stato del canale firmando questa proposta con la propria chiave privata. Berto riceve la proposta, ne verifica la validità ed eventualmente l'accetta inviando la proposta controfirmata ad Alice. Il pagamento è avvenuto, senza la necessità di ulteriori tempi di attesa o conferme.

Gli endpoint pubblici Ciascuna controparte di un payment channel mette a disposizione un server http. Gli endpoint pubblici sono detti /propose e /accept. L'endpoint /propose permette di ricevere una proposta di aggiornamento di bilancio. L'endpoint /accept permette di ricevere una proposta precedentemente inviata, controfirmata in Tabella 1.

Tabella 1: Struttura di una propose

Campo	Descrizione
seq	Il numero di sequenza
balance_a	Il balance di chi ha aperto il canale
balance_b	Il balance di chi ha effettuato il join del canale
sign	La firma della propose

Richiesta di chiusura Chiudere un canale significa aggiornare il balance on-chain delle parti in modo tale che corrisponda a quello dell'ultima propose comunemente accordata. La prima fase di questo processo è detta richiesta di chiusura. In particolare si porta in catena l'ultima propose comunemente firmata. In questo modo lo stato del canale passa da $ESTABLISHED$ a $CLOSED$. La richiesta di chiusura può essere effettuata da Alice o da Berto.

Finalizzazione della chiusura L'operazione di finalizzazione della chiusura viene effettuata da tutte e due le parti. Essa corrisponde al ritiro on-chain dei rispettivi fondi. Questa operazione può essere effettuata solo quando è passato un certo tempo dalla richiesta di chiusura. Il tempo che occorre attendere per finalizzare la chiusura è detto grace period.

Discutere una propose Alice (o Berto) potrebbe non comportarsi correttamente, portando in chiusura una propose diversa dalla più recente. In questo caso Berto può discutere la propose durante il grace period. Discutere una propose significa portare in catena una propose firmata da Alice con numero di sequenza maggiore rispetto a quella presentata. Nel caso in cui la discussione abbia successo, Alice viene punita; la punizione consiste nel trasferimento di tutti i suoi fondi a Berto.

Il problema della free-option Quando Alice invia una propose a Berto senza ricevere la controfirma, Berto ha il vantaggio di poter scegliere di

chiudere il canale con due propose, la penultima o l'ultima. Inviare una propose coincide con inviare un pagamento, quindi sebbene Berto possa decidere di presentare in catena la penultima propose, questa descriverà uno stato per lui più svantaggioso.

2.2.2 Inextinguishable payment channel

I payment channel permettono di trasferire un volume di coin limitato. Il valore trasferibile è fissato alla somma del balance di Alice e di Berto. Spesso questi canali sono sbilanciati, ovvero una delle due controparti effettua più pagamenti dell'altra. Un canale sbilanciato nel tempo prosciuga il balance di una delle due parti, rendendo il payment channel inutilizzabile. L'unica soluzione consiste nel chiudere il payment channel corrente e aprirne un nuovo, caricando nuovi fondi. Questa soluzione richiede delle operazioni on-chain onerose (deploy, apertura e join). Gli IPC (inextinguishable payment channel) superano questo problema, proponendo dei canali di pagamento che permettono di caricare e scaricare a caldo i balance.

Schema detach/attach Questo schema rappresenta un'estensione dello schema propose/accept. Esso permette di staccare un token off-chain e di attaccarlo on-chain. Un token rappresenta un certo quantitativo del bilancio. La struttura di un token è illustrato in Tabella 2.

Tabella 2: Struttura di un token

Campo	Descrizione
seq	Numero di sequenza del token
value	Valore del token
sign	Firma del token

Anche la struttura dati relativa a una propose viene estesa. I campi aggiunti sono illustrati in Tabella 3.

Tabella 3: Campi propose aggiuntivi in un IPC

Campo	Descrizione
hash token	L'hash relativo al token
type of propose	attach/detach

Ritiro a caldo Alice vuole ritirare a caldo 0.5 eth; effettua il detach off-chain di un token; invia a Berto una propose contenente un token di 0.5 eth che scala da balance_a. Berto risponde con propose e token firmati. Il token firmato rappresenta la PoD (Proof of Detachment). Alice effettua l'attach in catena della PoD e ritira a caldo 0.5 eth.

Ricarica a caldo Alice vuole ricaricare a caldo il canale di 0.5 eth; effettua l'attacch on-chain di un token depositando nello smart contract 0.5 eth. Questa operazione on-chain viene notificata a Berto; tale notifica rappresenta la PoA (Proof of Attachment). A questo punto Alice invia a Berto una propose in cui effettua l'attach di un token di pari valore e incrementa di 0.5 eth il proprio balance. Berto risponde con la propose firmata, confermando la ricarica a caldo.

Double spending di un token Quando Alice ritira a caldo presentando un token, lo smart contract associa una PoA (Proof of Attachment) relativa al numero di sequenza del token corrente. Questo permette allo smart contract di non accettare token già spesi.

2.3 Fulgur Hub

Sebbene i canali di pagamento e gli IPC rappresentino un punto di svolta per la scalabilità off-chain, essi sono degli strumenti rudimentali e con una esperienza utente limitata. Non è infatti pensabile dover inizializzare un canale di pagamento con ciascun individuo con cui si voglia instaurare un rapporto economico. Fulgur Hub nasce dalla necessità di migliorare l'esperienza utente degli IPC e di potenziare alcune delle loro caratteristiche.

2.3.1 Caratteristiche

Transazioni istantanee ed economiche In Bitcoin la conferma di una transazione richiede 60 minuti. In un IPC basta lo scambio di due messaggi su protocollo http per effettuare e confermare un pagamento. Questo apre diverse e interessanti prospettive economiche, ad esempio una macchina in cloud potrebbe essere pagata dopo ogni secondo di utilizzo o potremmo vedere il nostro stipendio accreditato dopo ogni minuto di lavoro; FulgurHub abilita questi casi d'uso.

Transazioni tra più di due entità In un IPC i pagamenti possono essere effettuati tra due partecipanti. FulgurHub consente di effettuare pagamenti tra gli N utenti registrati ad un FulgurHub.

Pagamenti ibridi FulgurHub permette di effettuare dei pagamenti ibridi. Ciascun utente infatti possiede due balance, uno on-chain e uno off-chain e può decidere di spostare dei fondi da uno stato off-chain a uno stato on-chain e viceversa. Inoltre abilita i pagamenti tra utenti di due FulgurHub diversi.

Autogestito In un IPC l'utente deve costantemente verificare e accettare la validità di un pagamento, oltre a contestare eventuali comportamentei scorretti della controparte. In FulgurHub i server degli utenti e dell'hub si

occupano di gestire autonomamente diversi scenari, limitando allo stretto necessario l'intervento manuale.

Pagamenti trustless Caratteristica essenziale è che un utente onesto abbia la certezza di non perdere i propri fondi. In sistemi centralizzati questa garanzia esiste perché ci si fida di un'entità centrale, come una banca o un servizio di e-payment. In un FulgurHub questa garanzia è data dal protocollo stesso, in questo senso i pagamenti sono trustless.

Passività e anonimato FulgurHub è un sistema passivo; questo significa che l'hub non contatta mai gli utenti, ma solo quest'ultimi contattano l'hub. Questo permette agli utenti di non dover fornire il loro indirizzo ip reale e quindi di poter effettuare pagamenti anche dietro una rete come Tor.

2.3.2 Lavori correlati

Tumblebit Si tratta di un hub di pagamenti anonimo basato su Bitcoin. L'approccio di centralizzazione garantisce anonimato e pagamenti trustless. Sfortunatamente il particolare payment channel adottato è unidirezionale e ha un tempo di vita limitato [5].

CoinBlesk Un bitcoin wallet che usa un server centrale che permette di eseguire dei pagamenti virtuali. Supporta micropagamenti istantanei, ma l'approccio non è considerabile trustless [2].

Lightning e Raiden Network Entrambi i network si basano su un grafo di payment channel bidirezionali. Un pagamento avviene in maniera analoga all'instradamento di un pacchetto su internet. Una volta trovato il percorso ottimo esso deve essere completato con successo in ciascun hop intermedio. Se un solo hop fallisce il pagamento fallisce. Questo garantisce l'atomicità dei pagamenti [10] [16]. Sebbene Lightning Network e Raiden Network siano progettati per essere decentralizzati, la realtà economica fa tendere la topologia

di rete alla centralizzazione; maggiore è il numero di hop, maggiori sono le commissioni e le probabilità di insuccesso. FulgurHub è stato disegnato con questo in mente e propone una topologia hub and spoke.

Capitolo 3

Analisi

3.1 Obiettivi

3.1.1 Dimostrazione di fattibilità

1. Transazioni OffChain-OffChain
2. Transazioni OnChain-OnChain
3. Transazioni OffChain-OnChain
4. Transazioni OnChain-OffChain
5. Prelievi a caldo
6. Ricariche a caldo

3.1.2 Dimostrare la scalabilità architetturale

3.2 Descrizione generale dell'architettura

3.2.1 Lo smart contract

3.2.2 Il client

3.2.3 L'hub

3.3 Casi d'uso

3.3.1 Apertura di un canale

1. Pre condizioni
2. Descrizione delle interazioni

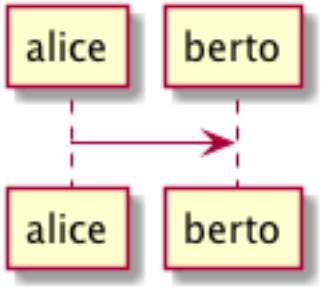

Figura 1: UML

3.3.2 Pagamento OnChain-OnChain

1. Pre condizioni
2. Descrizione delle interazioni
3. Gestione delle eccezioni
   (a) Credito insufficiente del client OnChain

3.3.3 Pagamento OffChain-OffChain

1. Pre condizioni
2. Descrizione delle interazioni
3. Gestione delle eccezioni
   (a) B non invia la ricevuta di pagamento ad A
   (b) Generazione di una miriade di token
   (c) L'hub non permette di attaccare un token
   (d) L'hub non permette di staccare un token
   (e) A si rifiuta di regolare un trasferimento nei confronti dell'hub
   (f) Tentativo di pagamento con un token scaduto
   (g) Mancanza di cooperazione nel ricevere un pagamento

3.3.4 Pagamento OffChain-OnChain

1. Pre condizioni

2. Descrizione delle interazioni

3.3.5 Pagamento OnChain-OffChain

1. Pre condizioni
2. Descrizione delle interazioni

3.3.6 Prelievo a caldo

1. Pre condizioni
2. Descrizione delle interazioni

3.3.7 Ricarica a caldo

1. Pre condizioni
2. Descrizione delle interazioni

3.3.8 Chiusura di un canale

1. Pre condizioni
2. Descrizione delle interazioni

3.3.9 Riscossione dei pending token

1. Pre condizioni

2. Descrizione delle interazioni
3. Gestione delle eccezioni
    (a) Tentativo di ritirare un pending token già usato

Capitolo 4

Progettazione e sviluppo

4.1 Le motivazioni tecnologiche

4.1.1 La blockchain: Ethereum

1. Supporto degli smart contract
2. Ambiente di sviluppo maturo
   (a) Solidity
   (b) Ganache
   (c) Web3
3. Altre soluzioni

   tesoz ecc..

4.1.2 Il linguaggio di programmazione: TypeScript

1. Supporto di web3

2. Tipizzazione forte

4.1.3 Il database lato server: Redis

1. Throughput considerevole in scrittura

2. Customizzazione delle qualità nei limiti del teorema CAP

   (a) Consistenza

   (b) Disponibilità

   (c) Sharding

4.1.4 Il database lato client: LevelDB

4.2 Lo smart contract

4.2.1 Implementazione in Solidity

4.2.2 Interfaccia in TypeScript

4.3 Il client

4.3.1 RPC privata

tabella riassuntiva degli endpoint

1. Join di un hub
2. Trasferimento OnChain-OnChain
3. Detach di un token OffChain-OffChain
4. Detach di un token OnChain-OffChain
5. Invio della PoD
6. Redimere un pending token
7. Attach di un token OnChain
8. Regolazione di un pagamento OffChain
9. Invio della ricevuta di pagamento

4.3.2 Endpoint pubblici

1. Ricezione di una PoD
2. Ricezione di una ricevuta di pagamento

4.3.3 Gestione degli eventi asincroni

1. Il monitor
2. Gli eventi
   (a) Detach di un token OnChain
   (b) Ricezione di una PoD

4.4 Hub

4.4.1 Endpoint pubblici

1. Ricezione di una propose
2. Ricezione di una ricevuta di pagamento

4.4.2 Gestione degli eventi asincroni

1. Il monitor
2. Gli eventi
   (a) Join di un utente

(b) Chiusura di un canale

(c) Ritiro di un pending token

Capitolo 5

Prove sperimentali

5.1 Gli obiettivi

5.1.1 Verifica delle performance delle transazioni Off-Chain

5.1.2 Verifica della scalabilità dell'hub

5.2 L'approccio adottato

5.2.1 Benchmark server

1. Deploy dell'ambiente di collaudo basato su Docker Swarm
2. Esecuzione del benchmark
   (a) Transazioni seriali

   (b) Transazioni concorrenti

   (c) Simulazione della latenza di rete

5.3 Il throughput lato client

5.3.1 Risultati

1. Al variare della RAM
   (a) Tabella
   (b) Grafico
2. Al variare della CPU
   (a) Tabella
   (b) Grafico

5.4 Il throughput lato hub

5.4.1 Risultati

1. Al variare della RAM
   (a) Tabella
   (b) Grafico
2. Al variare della CPU

(a) Tabella

5.5 Considerazioni sulle performance

5.6 Considerazioni sulla scalabilità

5.6.1 Replicare l'hub

5.6.2 Replicare redis

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Autogestione finanziaria dell'hub

Endpoint denominati in maniera diversa

[1] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. 2016. Cryptocurrencies without proof of work. In *International conference on financial cryptography and data security*, 142–157.

[2] Thomas Bocek, Sina Rafati, Bruno Rodrigues, and Burkhard Stiller. 2017. Coinblesk–a real-time, bitcoin-based payment approach and app. *Blockchain Engineering* (2017), 14.

[3] Vitalik Buterin and others. 2014. A next-generation smart contract and decentralized application platform. *white paper* (2014).

[4] Arthur Gervais, Ghassan O Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the security and performance

of proof of work blockchains. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 3–16.

[5] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In *Network and distributed system security symposium*.

[6] Sunny King and Scott Nadal. 2012. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August* 19, (2012).

[7] Jae Kwon. 2014. Tendermint: Consensus without mining. *Draft v. 0.6, fall* (2014).

[8] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 17–30.

[9] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[10] Joseph Poon and Thaddeus Dryja. 2016. The bitcoin lightning network: Scalable off-chain instant payments. *draft version 0.5* 9, (2016), 14.

[11] The bitcoin boom | the new yorker.

[12] Average number of transactions per block.

[13] Visa inc. Is a global payments technology company that connects consumers, businesses, financial institutions and governments in more than 200 countries and territories, enabling them to use electronic payments instead of cash and checks.

[14] A proof of stake design philosophy | vitalik buterin.

[15] State channels - an explanation.

[16] Raiden network.