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 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 [7].

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 [8].

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 [9].

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 [3]. 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 [10].

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 [4], [1], [5].
- **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 [6], [11].
- **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

Blockchain

Il problema La blockchain risolve il problema del double spending in un sistema peer-to-peer completamente decentralizzato [7]. 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 sono dei nodi della rete che si occupano di trovare un nonce cha faccia si che l'hash del blocco corrente abbia un numero D di zeri iniziali. 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 di questo termine in questo contesto è stato introdotto solo nel 2014, con la pubblicazione del whitepaper di Ethereum [2]. 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.

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.

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 singola operazione on-chain da ciascuna parte. L'operazione on-chain viene eseguita su uno smart contract dedicato al singolo payment channel. Questo singolo requisito garantisce di poter effettuare 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 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 in considerazione 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 2.1: Deploy on-chain dello smart contract di un payment channel.

Apertura Alice apre il canale. 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 $OPENED$.

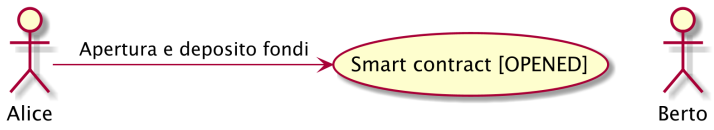

Figura 2.2: Apertura e deposito fondi on-chain in un payment channel.

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

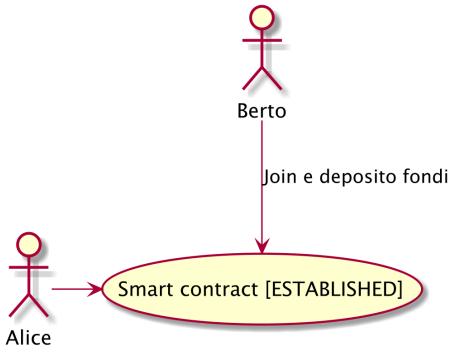

Figura 2.3: Join e deposito fondi on-chain in un payment channel.

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 parte di un payment channel mette a disposizione un server http. Gli endpoint pubblici sono /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. La struttura di una propose è illustrata nella tabella seguente.

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. Nel caso in cui la discussione abbia successo, Alice viene punita; la punizione consiste nel trasferimento di tutti i suoi fondi a Berto.

Inextinguishable payment channel

Le motivazioni di un IPC.

Schema attach/detach Descrizione di questo schema.

Estensione della propose La struttura dati relativa a una propose viene estesa, in particolare vengono aggiunti i sequenti campi:

Campo	Descrizione
$\text{hash}_{\text{token}}$	L'hash relativo al token
typeOfPropose	Il tipo di operazione che si sta effettuando (attach/detach)
typeOfChain	Il tipo di catena di riferimento (OnChain/OffChain)

Struttura di un token Un token ha la seguente struttura

Campo	Descrizione
seq	Numero di sequenza del token
value	Valore del token
expiration	Tempo di vita di un token
sign	Firma del token

Ritiro a caldo Effettua il ritiro a caldo effettuando prima il detach di un token OffChain e poi l'attach di un token OnChain.

Ricarica a caldo Senza schema attach/detach.

Fulgur Hub

Le motivazioni di Fulgur.

Obiettivi

Transazioni immediate Bitcoin ci mette un'ora, noi le vogliamo immediate.

Transazioni tra più di due entità Payment channel solo due, noi ne vogliamo N.

Transazioni intra hub Il sistema funziona anche con più di un hub.

Autogestito Non serve stare davanti al computer.

Non censurabile bla bla bla

Lavori correlati

Lightning Network Descrizione della topologia di rete a confronto e censura. Si predilige un hub ben interconnesso (+passaggi=+fee). Superamento del problema di ricerca del percorso ottimo.

NOCUST Conferma di una transazione non immediata.

Capitolo 3

Analisi

Obiettivi

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

Dimostrare la scalabilità architetturale

Descrizione generale dell'architettura

Lo smart contract

Il client

L'hub

Casi d'uso

Apertura di un canale

1. Pre condizioni
2. Descrizione delle interazioni

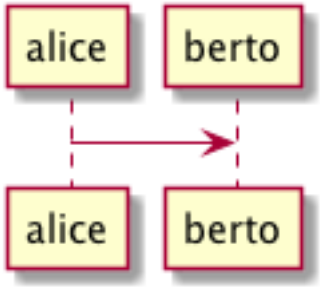

Figura 3.1: UML

Pagamento OnChain-OnChain

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

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

Pagamento OffChain-OnChain

1. Pre condizioni

2. Descrizione delle interazioni

Pagamento OnChain-OffChain

1. Pre condizioni
2. Descrizione delle interazioni

Prelievo a caldo

1. Pre condizioni
2. Descrizione delle interazioni

Ricarica a caldo

1. Pre condizioni
2. Descrizione delle interazioni

Chiusura di un canale

1. Pre condizioni
2. Descrizione delle interazioni

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

Le motivazioni tecnologiche

La blockchain: Ethereum

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

   tesoz ecc..

Il linguaggio di programmazione: TypeScript

1. Supporto di web3

2. Tipizzazione forte

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

Il database lato client: LevelDB

Lo smart contract

Implementazione in Solidity

Interfaccia in TypeScript

Il client

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

Endpoint pubblici

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

Gestione degli eventi asincroni

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

Hub

Endpoint pubblici

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

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

Gli obiettivi

Verifica delle performance delle transazioni OffChain

Verifica della scalabilità dell'hub

L'approccio adottato

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

Il throughput lato client

Risultati

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

Il throughput lato hub

Risultati

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

Considerazioni sulle performance

Considerazioni sulla scalabilità

Replicare l'hub

Replicare redis

Capitolo 6

Conclusioni e sviluppi futuri

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

[3] 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.

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

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

[6] 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.

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

[8] The bitcoin boom | the new yorker.

[9] Average number of transactions per block.

[10] 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.

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