Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

SIMULAZIONE DI ATTACCHI DI SICUREZZA SU TECNOLOGIE BLOCKCHAIN

Relatore:
Chiar.mo Prof.
Gabriele D'angelo

Presentata da:
Luca Serena

Corelatore:
Chiar.mo Prof.
Stefano Ferretti

III Sessione: 19/03/2020
Anno Accademico 2018/2019

Introduzione

Blockchain e criptovalute sono tra le innovazioni informatiche più discusse dell'ultimo decennio. Infatti non è necessario lavorare o essere competenti nel settore per aver sentito nominare questi termini e per sapere almeno approssimativamente il loro significato.
La blockchain è una struttura dati che funge da database distribuito di sola scrittura, nel quale i vari record vengono salvati sotto forma di transazioni, senza un'entità centrale a coordinare le operazioni.
A causa delle sue proprietà la struttura dati blockchain può essere impiegata in vari ambiti applicativi, ma il termine è principalmente associato al mondo delle criptovalute e in particolare al loro progenitore Bitcoin.
L'impatto delle criptovalute nell'economia non è particolarmente significativo attualmente, ma la sensazione è che possano aver aperto la strada a nuovi tipi di approcci per lo scambio telematico di denaro o di altri asset.
Anche i governi si sono accorti di questa novità informatica e hanno conseguentemente cominciato a legiferare sull'argomento e/o a pianificare l'utilizzo della blockchain per gestire il trattamento di dati pubblici o governativi.
Ci sono infatti diverse novità e aspetti tecnologici interessanti che caratterizzano i sistemi basati sulla blockchain, tra i quali sicuramente spiccano la decentralizzazione dell'informazione, l'uso di tecniche crittografiche per garantire la sicurezza e l'integrità dei dati, la possibilità di anonimizzare le transazioni e l'identità degli utenti e il fatto che non ci sia un ente centrale a controllare il valore delle monete.
Pur esssendo un'innovazione piuttosto recente gli studi sulle proprietà del-

la blockchain non mancano, e può essere utile capire quali siano i pregi e i limiti di questa tecnologia in particolare per quanto riguarda la sicurezza. Potrebbe esserci infatti la possibilità che uno o più attaccanti attuino un comportamento scorretto per avvantaggiarsi economicamente o per danneggiare un rivale. Come tutte le applicazioni informatiche quindi l'aspetto di sicurezza è fondamentale, anche al fine di garantire che ci sia la fiducia degli stackholder verso la tecnologia.
Spesso in un sistema complesso è difficile capire analiticamente il comportamento di tutte le componenti di fronte a determinati eventi o stimoli o in particolari situazioni. Può essere quindi vantaggioso costruire un modello, talvolta semplificato negli aspetti non rilevanti per lo scopo.
Nella blockchain infatti interagiscono una notevole quantità di attori e componenti singoli e quindi è un sistema che ben si presta a essere modellato in scala. Utilizzare un simulatore costruito ad hoc permette di semplificare alcuni aspetti del sistema per concentrarsi su quelli più critici e potenzialmente vulnerabili.
Nella tesi nello specifico si è scelto di utilizzare LUNES (e la sua variante LUNES-bitcoin), un software sviluppato da un gruppo di ricerca dell'Università di Bologna che permette di testare il funzionamento e le performance di protocolli complessi all'interno di reti peer-to-peer.

Il lavoro iniziale della tesi è stato quello di testare l'efficienza di alcuni protocolli per la disseminazione di messaggi (tra cui Dandelion, usato in Bitcoin per propagare le transazioni) nelle diverse configurazioni che può assumere una rete. Successivamente invece ho lavorato con Lunes-bitcoin, apportando alcune modifiche e miglioramenti che permettono una simulazione più realistica del comportamento della blockchain e degli attacchi informatici che possono essere effettuati su sistemi che usano tale struttura dati.

La prima parte di questa tesi è dedicata ad approfondire le tematiche collegate alla blockchain e alla simulazione di sistemi, nella seconda parte invece

viene presentato e analizzato il lavoro svolto su LUNES e LUNES-bitcoin. Nello specifico il documento è diviso nei seguenti capitoli:

- Blockchain. In questa sezione si parlerà della struttura dati che sta alla base della maggior parte delle criptovalute, descrivendo come generalmente viene implementata la blockchain, quali sono i principi che stanno alla base di questa tecnologia e con particolare riguardo ai vari meccanismi di validazione dei blocchi.

- Bitcoin. In questo capitolo si descrive nello specifico la strutturazione del sistema Bitcoin, con cenni anche alle possibili problematiche di sicurezza e ambientali.

- Simulazione di sistemi. In questa sezione viene descritto brevemente cosa si intende per simulazione e quali sono i possibili approcci per implementare un modello di un sistema. Viene poi descritto il funzionamento di LUNES e LUNES-bitcoin.

- Valutazione delle performance dei protocolli di gossip. Qui comincia la seconda parte della tesi. In particolare in questa sezione si discutono i risultati dei test effettuati su LUNES per valutare le performance dei vari protocolli di gossip nelle diverse configurazioni della rete. Si è testata inoltre l'incidenza che la scelta del protocollo di disseminazione delle transazioni può avere sull'esito dell'attacco di denial of service su Bitcoin.

- Quanto è realistico LUNES-bitcoin? In questo capitolo si discute su quanto sia realistico il modello proposto per simulare il sistema Bitcoin e si propongono alcuni miglioramenti.

- Forking. In questa ultima sezione vengono riproposti i test sugli attacchi informatici con la nuova configurazione di LUNES-bitcoin, ovvero quella in cui sono ammesse biforcazioni nella catena di blocchi.

Indice

1 Blockchain **1**

1.1 Funzioni di Hash 2

1.2 Sistemi centralizzati, decentralizzati e distribuiti 4

1.3 Utilizzi reali della blockchain 6

1.4 Merkle Tree 8

1.5 Blocchi 9

1.6 Modalità di validazione 11

1.6.1 Proof of work 13

1.6.2 Proof of Stake 17

1.6.3 Delegated proof of stake 18

1.6.4 Proof of Authority 19

2 Bitcoin **21**

2.1 Transazioni 23

2.1.1 Script, un linguaggio per transazioni 24

2.1.2 Transazioni Pay-to-PubKey-Hash 26

2.1.3 Transazioni Pay to Script Hash 29

2.1.4 NULL DATA 30

2.2 Topologia della rete 31

2.3 Blocchi 33

2.4 Attacchi alla sicurezza 34

2.5 Aspetti economici 37

3 Simulazione di sistemi **41**
3.1 LUNES 45
3.2 LUNES-bitcoin 48

4 Valutazione delle performance dei protocolli di gossip **53**
4.1 Analisi sui protocolli di gossip 63

5 Quanto è realistico LUNES-bitcoin? **69**
5.1 Hashrate dell'attaccante e mining pool 69
5.2 Rapporto tra transazioni e blocchi 73
5.3 Durata di uno step della simulazione 74
5.4 Catena unica 75
5.5 Fee 76

6 Forking **77**
6.1 Selfish mining 83
6.2 51% 88

Conclusioni **93**

Bibliografia **97**

Capitolo 1

Blockchain

La blockchain è una struttura dati che consente la creazione e la gestione di database non relazionali di grandi dimensioni, i cui record sono distribuiti tra i vari nodi della rete. Per inserire nuovi dati gli utenti eseguono delle transazioni, che si avvalgono di tecniche criptografiche per far sì che sia garantita l'autenticità dei messaggi.
L'informazione presente nella blockchain viene inserita all'interno di una sequenza di blocchi, che sono essenzialmente dei contenitori di un numero variabile di dati. I blocchi sono composti da una lista di transazioni e da uno header, che contiene varie informazioni tra cui quelle che consentono a un qualsiasi utente del sistema di assicurarsi della validità dei dati.
Tutti i blocchi sono logicamente collegati tra di loro e tranne il primo blocco della catena tutti hanno un riferimento a uno e un solo blocco precedente.
La blockchain punta quindi a creare un database che rispetti le seguenti proprietà:

- Sicurezza, garantita grazie a tecniche crittografiche come le funzioni di hash. Qualsiasi utente può verificare istantaneamente la validità dei dati di un blocco e un tentativo di falsificare le informazioni in esso contenute si ripercuoterebbe su tutti i blocchi successivi.

- Immutabilità, le transazioni una volta inserite non possono essere annullate o modificate.

- Trasparenza, i dati sono salvati in un ambiente distribuito e sono perciò visibili a tutti. Infatti ogni nodo attivo nella rete possiede teoricamente tutti i dati del ledger.

- Convenienza. A differenza delle transazioni convenzionali non è più necessaria la presenza di una banca o di un'entità terza che funga da intermediario.

1.1 Funzioni di Hash

Una funzione di hash è una funzione one way che mappa dei dati di lunghezza arbitraria in una stringa di lunghezza fissa chiamata hash o digest.
Queste funzioni si rivelano utili in applicazioni come la firma digitale perché consentono di provare l'autenticità di un messaggio. Spesso sono usate in combinazione con tecniche di cifratura asimmetrica che invece garantiscono la confidenzialità della comunicazione.
Il ricevente di un messaggio per assicurarsi della sua autenticità calcola la funzione di hash sui dati ricevuti, e poi confronta l'output con il digest fornitogli. Se i due valori non corrispondono allora significa che i dati sono stati manomessi o che comunque non sono arrivati integri.
Per soddisfare i requisiti di sicurezza richiesti è essenziale che vengano rispettate le seguenti proprietà:

- Determinismo. Ogni funzione di hash applicata allo stesso input deve resituire sempre lo stesso output.

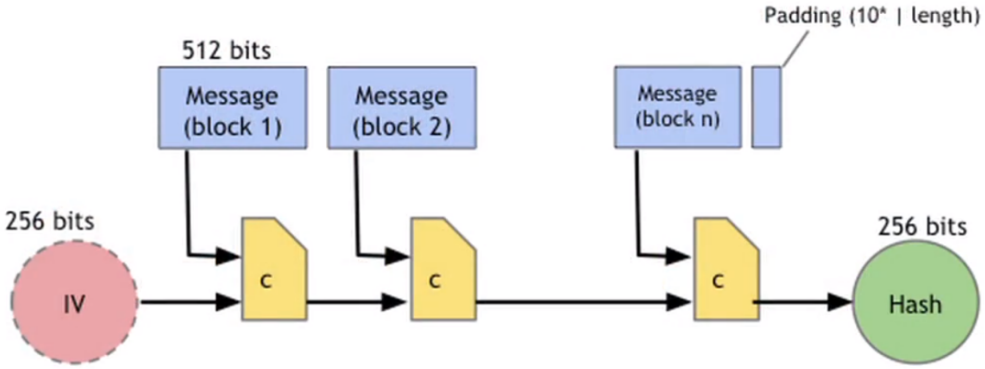
- Non invertibilità. A partire da un digest non deve essere possibile ottenere alcuna informazione riguardo al messaggio originale.

- Effetto cascata. Il cambiamento di anche un solo carattere della stringa in input deve rendere l'output completamente differente.

- Resistenza alle collisioni. Deve essere estremamente improbabile ottenere delle collisioni, ovvero trovare due messaggi di senso compiuto che generano lo stesso digest.
- Veloce da calcolare. La procedura di calcolo di un digest a partire da qualsiasi dato deve essere veloce per qualsiasi tipo di input.

La maggior parte delle blockchain usa la funzione di hash Secure Hash Algorithm 256 (SHA-256)[22], che restituisce un digest di 8 byte. Il funzionamento di SHA-256 può essere descritto nella seguente maniera:

1. Vengono aggiunti dei bit di imbottitura all'input in modo da ottenere un messaggio che risulto congruo a 448 modulo 521
2. Si aggiunge alla sequenza ottenuta un intero unsigned di 64 bit che rappresenta la lunghezza del messaggio originale.
3. Vengono creati 8 buffer da 32 bit e inizializzati con valori esadecimali
4. La sequenza di bit "messaggio + imbottitura + lunghezza del messaggio" viene divisa in blocchi da 512 bit.
5. Si esegue la funzione di compressione vera e propria composta da 4 ciclo di 20 passi ciascuno. In ogni ciclo, pur utilizzando diverse funzioni logiche primitive, l'algoritmo prende in input i valori dei blocchi, i buffer e una costante. Al termine dell'ultima iterazione il valore dei registri costituirà l'output della funzione di hash.



Essendo la dimensione del digest di 32 byte, il numero di possibili valori che si possono ottenere applicando la funzione di hash è di 2^{256}, una cifra simile alla stima del numero di atomi presenti nell'universo conosciuto. Generare collisioni è quindi estremamente improbabile. È anche garantito l'effetto cascata visto che input simili generano sempre hash molto diversi tra loro.

Ad esempio applicando SHA-256 alla stringa "Messaggio di prova" si ottiene il seguente digest:

6b5638208ce6ac3047e6d9fb0087cd69b9f15376c514598b328ceeb6bf917969

Applicando la stessa funzione di hash dopo aver aggiunto un punto alla fine della stringa in input invece si ottiene:

f5b1c9b64b6535881eabd84f424d7d2edaa04e1fef2836e8db8e0464a83e709ec

Si noti come l'hash di "Messaggio di prova." sia completamente differente nonostante ci sia stata l'aggiunta di un solo carattere.

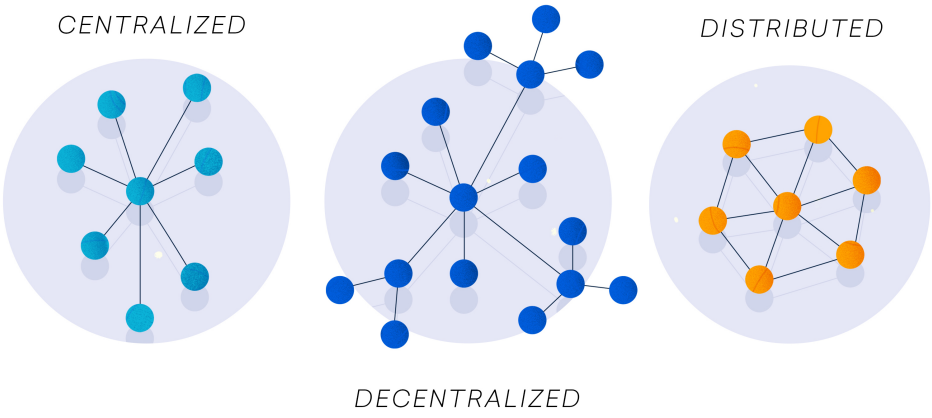
1.2 Sistemi centralizzati, decentralizzati e distribuiti

Uno dei tratti distintivi della blockchain è il fatto che le informazioni sono distribuite. Gli aggiornamenti del database non provengono da un'entità centralizzata ma sono comunicati dagli altri peer del sistema. Normalmente nelle applicazioni informatiche in cui è necessario accedere a dei dati attraverso la rete si fa uso di modelli centralizzati. Nella classica architettura client-server infatti ogni terminale si connette a un server per poi richiedere le informazioni di cui ha bisogno. È il modello più diffuso su internet ed è

usato ad esempio per accedere al contenuto HTML delle pagine web.
I sistemi centralizzzati presentano le seguenti caratteristiche:

- Presenza di un clock globale, il sistema è costituito da un solo server/master e da tanti client/slave che si sincronizzano col clock globale dell'entità centrale.
- Presenza di una singola entità centrale che coordina tutte le altre.
- Single point of failure, nel caso si riscontrino problemi nell'entità centrale tutto il resto del sistema non sarà in grado di funzionare autonomamente.

Se il modello centralizzato propone una logica "uno a molti", nei modelli decentralizzati questa logica si ripropone a livello locale, con tanti soggetti centrali che fungono da server. In questo caso non ci sarà una singola entità a possedere e diffondere l'informazione ma ce ne saranno diverse. Con questa architettura non si pone più il problema del single point of failure in quanto un guasto a uno dei soggetti centrali della rete può essere mitigato dalla presenza delle altre entità con ruolo analogo.
Nei sistemi distribuiti invece non c'è una singola entità che coordina l'azione degli altri nodi. L'informazione è accessibile a chiunque e i meccanismi di consenso e validazione dei dati non si basano sull'autorevolezza dell'entità centrale.



La blockchain è un tipo di sistema distribuito: ogni nodo della rete possiede, comunica e può richiedere le informazioni presenti nel sistema e vi sono inoltre meccanismi di consenso che permettono di verificare l'integrità dei dati.

1.3 Utilizzi reali della blockchain

La blockchain è una struttura dati che è principalmente nota per essere la tecnologia utilizzata per tenere traccia degli scambi di denaro digitale effettuati con Bitcoin o con altre criptovalute. Questa non è però l'unica applicazione possibile, in quanto la tecnologia può essere sfruttata in diversi settori, come quello alimentare, quello sanitario, quello immobiliare, ecc...
I record contenuti nei blocchi non necessariamente contengono solo dati ma possono anche essere degli script più o meno complessi. A volte il codice è minimale e serve solo a specificare il destinatario, l'ammontare da trasferire e a provare l'identità e la disponibilità economica del mittente. In altri casi invece si tratta di script scritti in linguaggi di programmazione Touring-completi, che consentono l'esecuzione dei cosiddetti smart contract[26].
Gli smart contract sono contratti digitali, che si differenziano da quelli tradizionali per avvelersi della firma digitale e per non essere scritti in linguaggio discorsivo ma bensì in codice. Per questo scopo si usano infatti linguaggi orientati ai contratti come Solidity o Liquidity. Con gli smart contract è possibile scambiare denaro, azioni, proprietà e asset di vario genere. I principali vantaggi di questa teconlogia sono:

- Velocità. Le transazioni vengono diffuse immediatamente.

- Sicurezza e trasparenza. Il contenuto del contratto è criptato e quindi l'accesso alle informazioni è garantito solo alle parti coinvolte.

- Efficienza. Permette di eludere certi aspetti burocratici e rende non più necessaria la presenza di diversi documenti cartacei.

- Accuratezza. È richiesto di esplicitare tutti i termini e le possibili casistiche del contratto in modo da evitare errori che possono accadere con i contratti tradizionali.

Gli svantaggi dell'uso di smart contract risiedono invece principalmente nel fatto che viene richiesta una conoscenza informatica da parte di chi li scrive, e non sempre i costi derivanti dall'uso di smart contract sono minori rispetto a quelli che si otterrebbero seguendo le procedure tradizionali. Inoltre il codice inserito nella blockchain è irreversibile e quindi non è possibile correggere eventuali bug e inconsistenze, che possono invece essere sfruttati da eventuali attaccanti.
Bitcoin, lanciato nel 2009, è considerato il primo sistema per lo scambio di criptovalute in modo libero e decentralizzato. Da allora sono nate oltre 6000 criptovalute, tra le quali si sta distinguendo Ethereum, che attualmente è probabilmente il concorrente più famoso di Bitcoin. Ethereum, lanciato nel 2015, è un sistema pensato appositamente per utilizzare gli smart contract.

La sempre maggiore diffusione di tecnologie legate alla blockchain ha portato diverse nazioni a legiferare sull'argomento. In Italia il 23 gennaio 2019 è stato approvato un emendamento al Senato che include formalmente per la prima volta nel nostro ordinamento la tecnologia della blockchain e gli smart contract. In particolare nell'articolo 8 bis vengono definite le "tecnologie basate su registri distribuiti e smart contract" nella seguente maniera: [8]

1. Si definiscono "tecnologie basate su registri distribuiti" le tecnologie e i protocolli informatici che usano un registro condiviso, distribuito, replicabile, accessibile simultaneamente, architetturalmente decentralizzato su basi crittografiche, tali da consentire la registrazione, la convalida, l'aggiornamento e l'archiviazione di dati sia in chiaro che ulterior-

mente protetti da crittografia verificabili da ciascun partecipante, non alterabili e non modificabili.

2. Si definisce " smart contract" un programma per elaboratore che opera su tecnologie basate su registri distribuiti e la cui esecuzione vincola automaticamente due o più parti sulla base di effetti predefiniti dalle stesse. Gli smart contract soddisfano il requisito della forma scritta previa identificazione informatica delle parti interessate, attraverso un processo avente i requisiti fissati dall'Agenzia per l'Italia Digitale con linee guida da adottarsi entro 90 giorni dall'entrata in vigore della legge di conversione del decreto-legge.

3. La memorizzazione di un documento informatico attraverso l'uso di tecnologie basate su registri distribuiti produce gli effetti giuridici della validazione temporale elettronica di cui all'articolo 41 del Regolamento UE n. 910/2014.

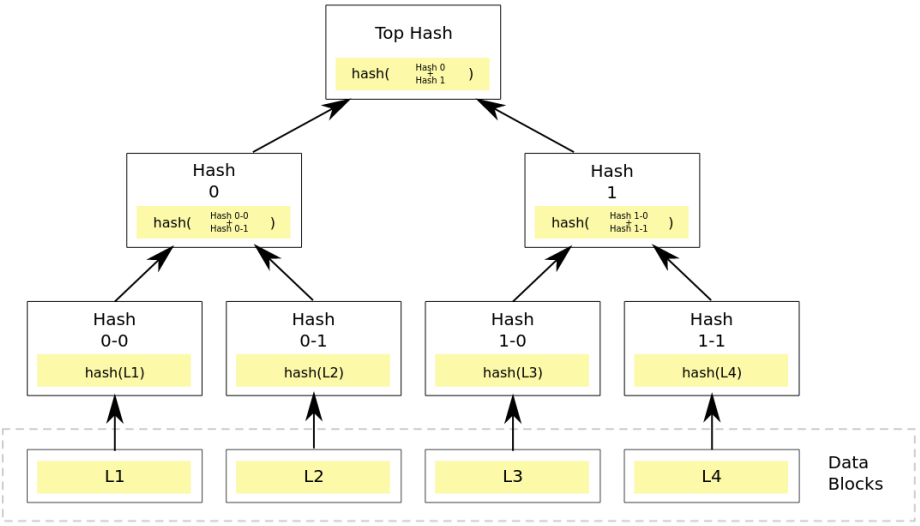
4. Entro 90 giorni dall'entrata in vigore della legge di conversione del decreto-legge, l'Agenzia per l'Italia Digitale individua gli standard tecnici che le tecnologie basate su registri distribuiti debbono possedere ai fini della produzione degli effetti di cui al comma 3.

L'aspetto innovativo dell'emendamento è che attribuisce valore giuridico alle transazioni sulla blockchain e sancisce che uno smart contract può assumere un valore giuridico al pari di un contratto scritto in forma tradizionale, attribuendogli anche valenza probatoria.

1.4 Merkle Tree

Quando le transazioni vengono inserite nei blocchi vengono solitamente organizzate ad albero binario, in particolare in una struttura dati appositamente pensata per la blockchain chiamata Merkle tree[18].
Nei Merkle tree le foglie sono etichettate con dei dati mentre i nodi non foglia

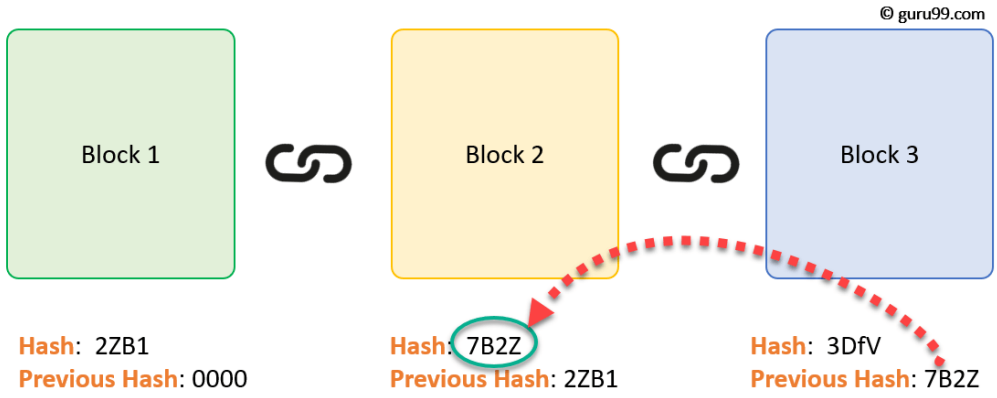
sono etichettati con l'hash della concatenazione del contenuto dei nodi figli. Procedendo bottom-up si arriva fino ad avere un singolo hash, il Merkle root, da cui un utente può verificare l'integrità del contenuto di tutte le transazioni.



Nella figura di esempio le foglie contengono i dati (L1, L2, L3, L4) mentre i nodi al penultimo livello contengono l'hash dei dati presenti nelle rispettive foglie. I nodi al Livello 1 contengono rispettivamente l'hash della concatenzione dell'hash di L1 con l'hash di L2 e l'hash della concatenazione dell'hash di L3 con l'hash di L4. Al livello 0 è presente l'hash root.

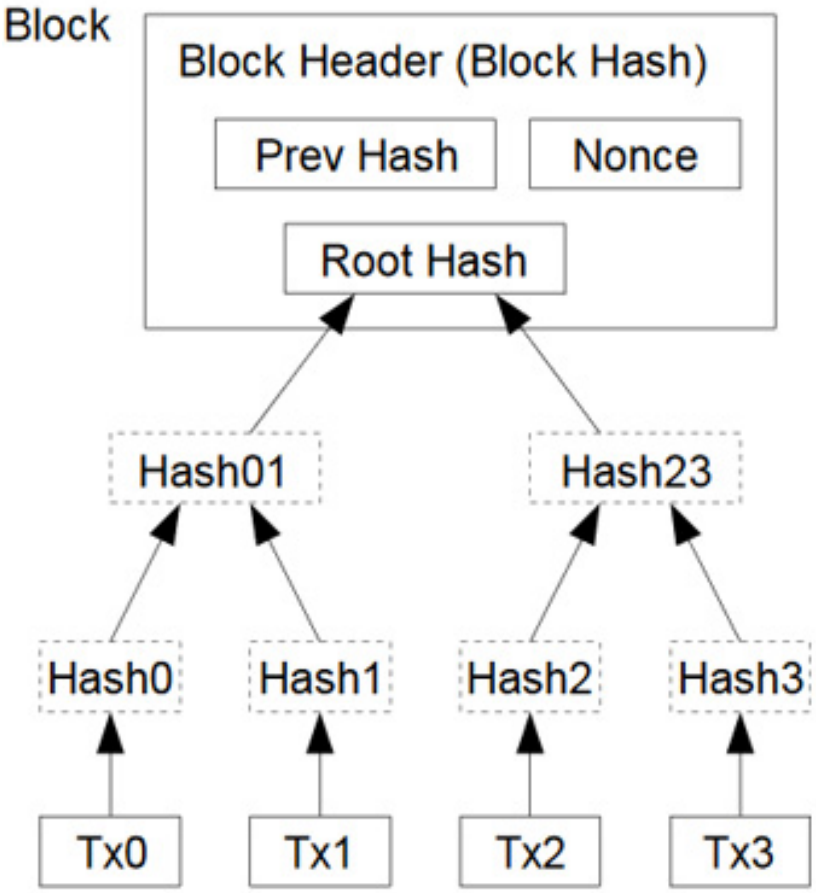
1.5 Blocchi

La blockchain è costituita da un insieme di blocchi e ognuno di questi ha un riferimento al blocco che lo precede, permettendo la ricostruzione di una catena logica. È possibile che si verifichi il caso in cui più di un blocco fa riferimento allo stesso blocco precedente, soprattutto nei sistemi come Bitcoin in cui la creazione di blocchi avviene in maniera concorrente tra i vari nodi della rete. In questo caso si parla di forking, cioè si diramano più catene a partire da uno stesso nodo.



In genere il meccanismo attuato per mettere d'accordo i vari nodi della rete su quali siano i blocchi da considerare validi è prendere in considerazione la lunghezza delle catene. La catena di lunghezza maggiore è considerata quella di riferimento. Quando si crea un nuovo blocco nel database distribuito, questo viene inserito in testa alla catena attualmente più lunga della blockchain. I blocchi che si trovano nelle altre biforcazioni vengono ignorati e le transazioni contenute in essi non sono considerate come effettivamente eseguite.

I blocchi sono costituiti da una lista di transazioni e da uno header. Quest'ultimo contiene l'hash del blocco precedente (grazie al quale è possibile ricostruire la catena), l'hash del Merkle root (grazie al quale è possibile provare la validità di tutte le transazioni contenute nel blocco), marche temporali e altre informazioni che consentono di assicurarsi della validità di un blocco e di stabilire con esattezza la versione del protocollo utilizzata.

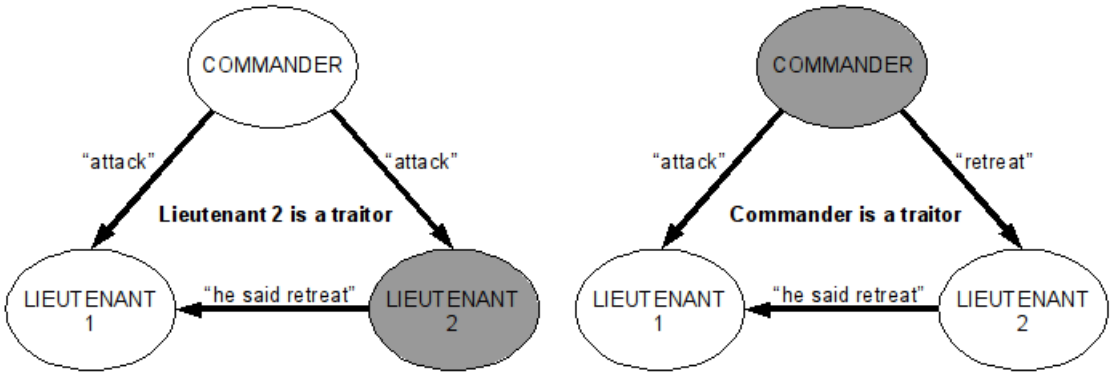


Da un punto di vista logico quindi un blocco potrebbe quindi avere un aspetto simile a quello della figura appena sopra: le transazioni sono organizzate nel Merkle tree e solo l'hash della radice dell'albero è presente nello header.

1.6 Modalità di validazione

Uno dei punti di forza della blockchain è la capacità di offrire buone soluzioni per il problema dei generali bizantini, ovvero su come raggiungere il consenso tra le varie componenti di un sistema distribuito nel caso in cui siano presenti informazioni discordanti. Questo problema prende il nome dall'allegoria con cui per la prima volta dei ricercatori esemplificarono questo grattacapo[23].

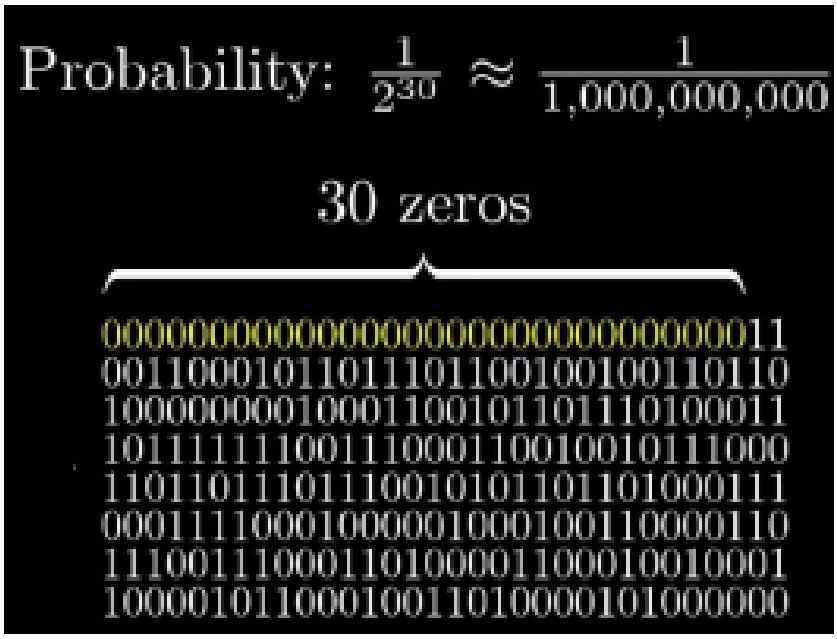
Nell'esempio diverse truppe ubicate in postazioni differenti circondano un castello che si vuole espugnare e ogni truppa è guidata da un generale. Ci può essere esito positivo da parte degli invasori solo in caso di un attacco simultaneo, perciò i generali devono coordinarsi tramite messaggi per concordare se agire o se ritirarsi. Uno o più generali potrebbero però essere dei traditori e mandare appositamente messaggi sbagliati per confondere gli altri generali, che riceveranno così informazioni discordanti su quale sia la decisione comune da prendere. Nel caso il numero di malintenzionati raggiunga o superi un terzo del totale degli attori non è possibile raggiungere il consenso. Si consideri la situazione in cui il comandante supremo A impartisce un ordine ai suoi due generali B e C. Nell'esempio B è fedele al suo condottiero, mentre C è un traditore. A consegna ad entrambi il messaggio chiedendo di attaccare o di ritirarsi e C inoltrerà a B l'informazione opposta a quella ricevuta. In questo modo B non sarà in grado di scegliere quale azione intraprendere avendo ricevuto comunicazioni discordanti, come nella figura in basso a sinistra.
È possibile anche che sia il comandante stesso a essere il traditore, e in questo caso sarà direttamente A a mandare informazioni contrastanti ai due generali, rendendo a entrambi impossibile acconsentire su quale sia la decisione comune da intraprendere (figura a destra).



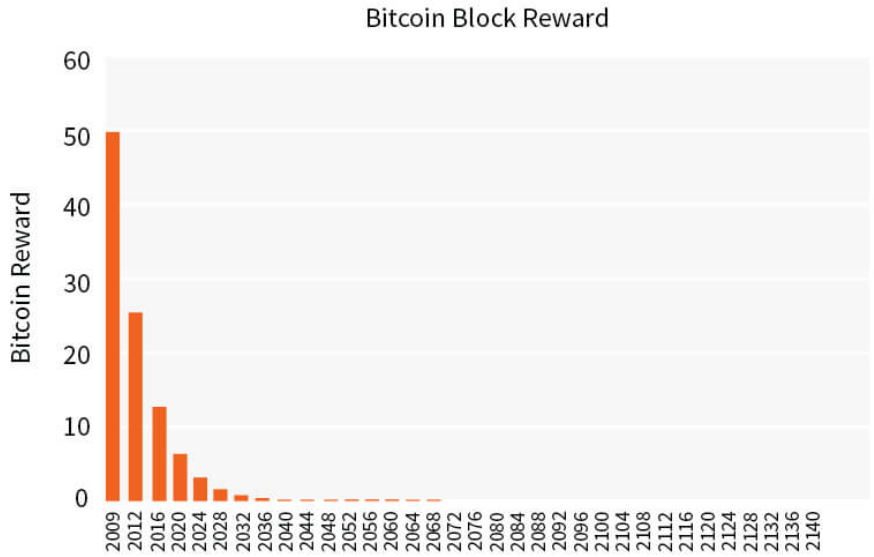
Firmando i messaggi in modo sicuro (con tecniche di crittografia asimmetrica e firma digitale) però è possibile raggiungere il consenso anche di fronte a un numero di traditori maggiore di un terzo del totale. In particolare nella blockchain ci sono diversi meccanismi per raggiungere il consenso tra i vari nodi del sistema e le metodologie più usate sono proof of work e proof of stake.

1.6.1 Proof of work

Il proof of work è il meccanismo di validazione dei blocchi più diffuso tra le criptovalute. In questo sistema diversi attori, chiamati miner o minatori, competono tra di loro per trovare la soluzione di un puzzle crittografico. Il primo miner ad avere successo comunica il blocco alla rete, validando così le transazioni inserite e ottenendo una ricompensa economica. I miner di Bitcoin guadagnano sia dalle fee dei mittenti delle transazioni (ovvero una quantità di denaro che gli esecutori del pagamento virtuale donano al miner per incentivarlo a inserire la transazione in un blocco valido) sia da una ricompensa che il sistema dona a chi riesce a risolvere il puzzle computazionale. Il proof of work è usato ad esempio da Bitcoin, Litecoin, Dogecoin, Monero, ecc...
Nello header dei blocchi dei sistemi che si avvalgono della proof of work sono inseriti due campi per gestire il mining, *nonce* e *target*. Il minatore, per validare un blocco, prova a indovinare un valore del nonce tale che applicando una funzione di hash sul blocco venga prodotto un digest che inizia con un numero di zeri maggiore o uguale al valore indicato dal target.



La precedente figura mostra l'esempio in cui il valore del campo *target* è 30, ed è quindi necessario trovare una configurazione del blocco tale per cui il suo hash cominci per almeno 30 zeri, cosa che avverrà mediamente ogni $2^{30} = 1.073.741.824$ tentativi.
Nel caso nessuna delle 2^{32} combinazioni del nonce offra una soluzione valida si potrà procedere aggiornando la validazione temporale del blocco, operazione che consente di effettuare nuovi 2^{32} tentativi.
La ricompensa per aver trovato una soluzione ammissibile di un blocco però non è fissa, ma è destinata a calare nel tempo fino a tendere a 0. Per questo motivo oltre una certa data non sarà più possibile guadagnare da questo meccanismo e i futuri minatori si dovranno sostentare esclusivamente sulle fee dei mittenti delle transazioni.
Precisamente l'ammontare della ricompensa viene dimezzata ogni 210000 blocchi minati, partendo da un valore inziale di 50 bitcoin. Dopo 32 dimezzamenti il valore della ricompensa sarà impostato a 0. Nel 2020 è previsto il terzo dimezzamento del numero di criptovalute generate in seguito alla genesi di un blocco, col valore della ricompensa che diventerà di 6.25 bitcoin.

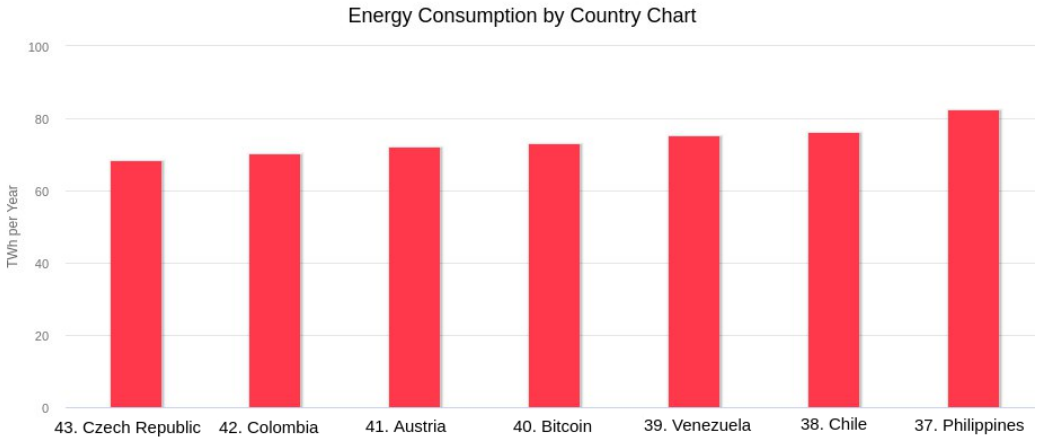


La difficoltà del puzzle crittografico viene aggiornata ogni due settimane[11], cambiando dinamicamente in base alla potenza computazionale dell'insieme dei miner. L'obiettivo è quello che venga minato in media un blocco ogni 10 minuti.

Negli ultimi anni la quantità di risorse impiegate per il mining è salita drasticamente. A eseguire i tentativi di minare un blocco sono solitamente elaboratori costruiti appositamente per tale scopo e che hanno un hardware ottimizzato per eseguire la funzione SHA-256 il più velocemente possibile. Chi investe in risorse hardware per minare i blocchi spesso non si mette in proprio, ma si aggrega a uno dei cosiddetti pool, ovvero gruppi di minatori che uniscono le loro risorse per risolvere il puzzle crittografico e dividersi gli eventuali guadagni in base alla potenza di calcolo offerta.

Da un lato la crescente competizione per minare i blocchi garantisce una maggiore sicurezza, rendendo per un eventuale malintenzionato quasi impossibile raggiungere una potenza di calcolo tale da poter portare sistematicamente a compimento attacchi contro il sistema. Dall'altro lato però causa un grandissimo consumo di energia elettrica e conseguentemente fa sì che Bitcoin

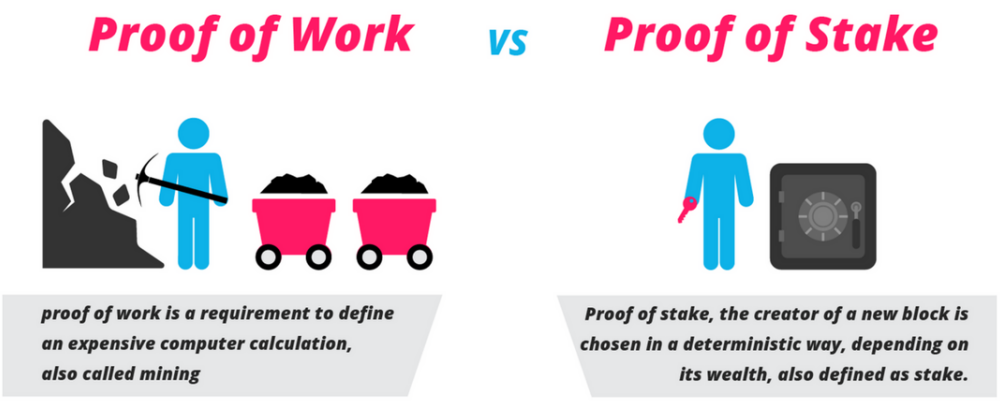
sia una fonte di inquinamento non trascurabile. Un esempio significativo è il fatto che con i sistemi VISA, per attuare tutte le operazioni per il funzionamento e il mantenimento del sistema, vengono consumati mediamente 151 KiloWatt all'ora per 100000 transazioni, mentre in Bitcoin vengono impiegati 654 KiloWatt all'ora per una singola transazione.
Secondo i dati di ottobre 2019 [6] il consumo di energia elettrica causato da Bitcoin è di 73.12TWh, quantità che, se Bitcoin fosse considerato come uno stato nazionale, lo collocherebbe al quarantesimo posto per consumo di elettricità.



È complesso, pur conoscendo la quantità di energia consumata, fare una stima accurata dell'inquinamento prodotto direttamente dall'attività di mining. Nei sistemi come Bitcoin gli attori della rete sono anonimi, per cui è difficile geolocalizzare i minatori e conseguentemente capire come sia stata prodotta l'energia elettrica utilizzata. Si pensa però che la maggior parte (circa il 60% [7]) della potenza di calcolo per il mining provenga dalla Cina, nazione in cui genericamente ci si affida al carbone per generare elettricità. Questo significa che l'inquinamento dovuto al mining è particolarmente significativo, nonostante molti complessi di mining situati in Europa e nel continente americano usino fonti di energia rinnovabili.

1.6.2 Proof of Stake

Un'altra maniera per raggiungere il consenso distribuito è la cosiddetta proof of stake. In questo caso il validatore del blocco non sarà un minatore che ha appena risolto un puzzle crittografico ma bensì un utente del sistema scelto casualmente. Questa selezione non sarà però totalmente randomica ma proporzionale alla quantità di denaro che l'utente ha depositato come deposito di garanzia. Se il validatore si comporterà in modo corretto nel processo di validazione dei blocchi riceverà una ricompensa, altrimenti perderà il denaro depositato. Più questa quota è alta più sarà nell'interesse dell'utente comportarsi onestamente e quindi questo verrà scelto con probabilità maggiore come prossimo validatore. Questa condizione pone però il problema, difficilmente risolvibile con questo meccanismo, del rich get richer, ovvero il fatto che il sistema favorisce il guadagno degli utenti che già possiedono tanti asset digitali.



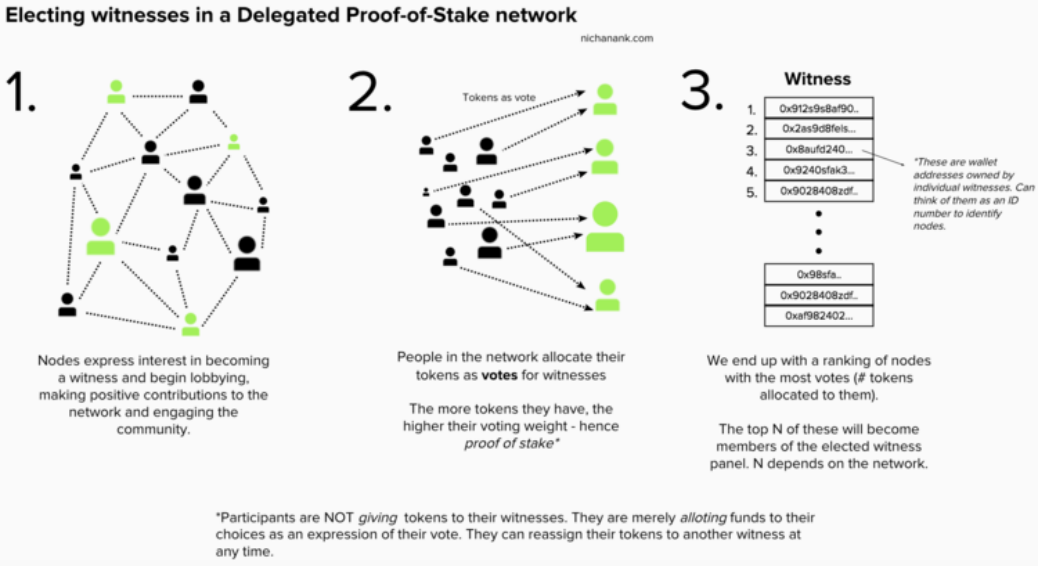
Una variante della proof of stake descritta precedentemente è quella basata sull'età, in cui non conta solo la quantità di denaro depositata ma anche il numero di giorni in cui le monete sono state possedute. PeerCoin è una criptovaluta digitale che utilizza questo meccanismo di validazione [19].
Per minimizzare il problema del rich get richer si fa in modo che quando una quantità elevata di denaro viene utilizzata per firmare il blocco, tali monete

abbiano poi il valore di anzianità reimpostato a 0. Si dovrà attendere almeno un mese prima che si possa di nuovo utilizzare quei soldi per la validazione di un altro blocco.
La validazione tramite la proof of stake è sicuramente più ecologica, in quanto la quantità di energia elettrica utilizzata per aggiungere un blocco alla blockchain è trascurabile.
Riguardo alla sicurezza ci sono dei pro e dei contro, i fautori della PoS sostengono che questo meccanismo sia più sicuro perché acquistare una certa percentuale di monete del sistema è tendenzialmente più costoso che acquisire l'analoga percentuale di potenza computazionale tra i miner di un sistema che usa la proof of work.

1.6.3 Delegated proof of stake

Delegated proof of stake (DPos) è una variante del PoS, introdotta per la prima volta nel 2013 nella blockchain di BitShares[20]. La tecnologia si basa sull'elezione democratica da parte degli stakeholder di alcune figure chiave (testimoni e delegati) per il funzionamento del sistema. Ogni stakeholder può votare un numero di volte proporzionale alla quantità di asset posseduti, o in alternativa può delegare la sua decisione a un altro utente, che quindi voterà a suo nome. In questo modo ci si assicura che non vengano creati un grande numero di profili falsi al solo scopo di influenzare l'esito delle elezioni.
Il ruolo di testimone e delegato può differire a seconda delle implementazioni, e non necessariamente le due funzionalità sono distinte. In Bitshares ad esempio i testimoni si occupano della creazione e della validazione dei blocchi. I delegati invece fanno da supervisori del funzionamento della blockchain e, pur non avendo nessun ruolo nella produzione di blocchi, possono proporre alcune modifiche alla configurazione del sistema come il cambio della dimensione dei blocchi o della ricompensa erogata per la validazione di un blocco. Quando viene proposta una modifica questa viene sottoposta alla votazione degli utenti per stabilire se entrerà o meno in vigore.
I testimoni si alternano per la validazione dei blocchi, che solitamente avvie-

ne ogni pochi secondi. Se un testimone si comporta in maniera inadeguata o fallisce costantemente nell'attività di validazione allora avrà danni reputazionali e sarà agevolmente espulso e sostituito da un nuovo testimone. È possibile inoltre che, analogamente al proof of stake, venga chiesto ai validatori di depositare una certa quantità di criptovalute come fondo di garanzia, da confiscare nel caso di comportamenti malevoli.



1.6.4 Proof of Authority

Proof of Authority (PoA) [21] è un algoritmo di consenso basato sulla reputazione che introduce una soluzione efficiente e pratica per la validazione dei blocchi. Può essere vista come una variante del proof of stake in cui però la garanzia di un comportamento onesto non è data dalla quantità di soldi depositati dal validatore ma dalla sua autorevolezza. Infatti in questo caso gli attori che si occupano di inserire i nuovi blocchi nella blockchain non sono entità anonime o sconosciute ma sono utenti fidati e la cui identità è pubblica e svolgono la funzione di moderatori del sistema. I validatori in questo caso eseguono software che in modo automatizzato inserisce le transazioni nei

blocchi senza il bisogno di monitorare costantemente l'attività di validazione. Sarà nell'interesse del validatore quindi comportarsi onestamente e assicurarsi che il software in esecuzione sul suo computer non venga compromesso, in modo da poter continuare a essere nella ristretta comunità degli utenti fidati del sistema.

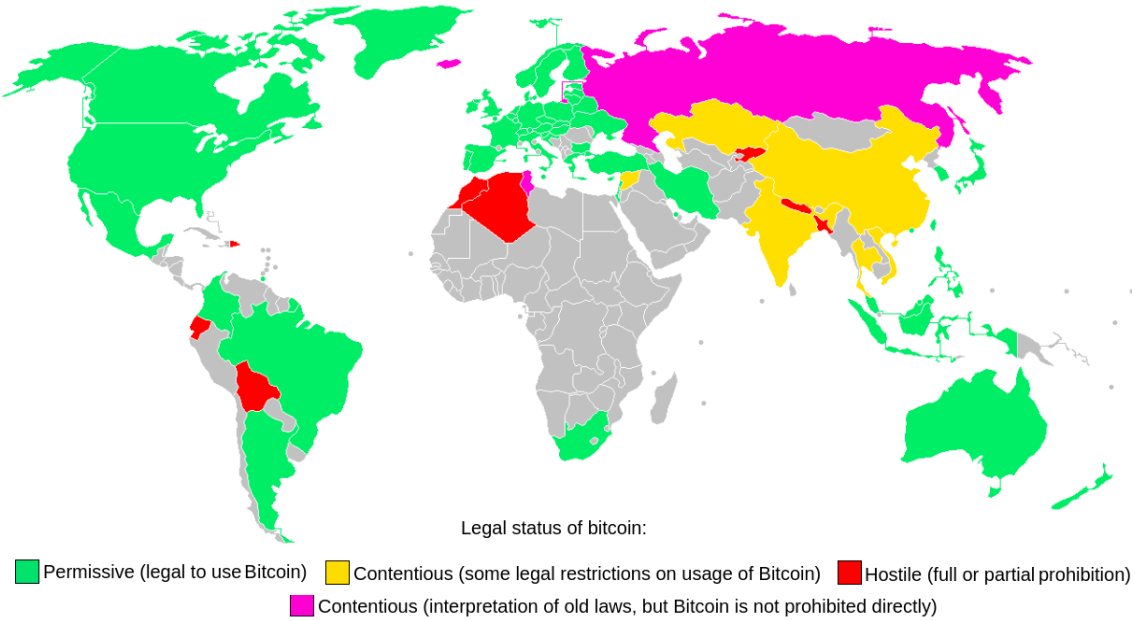
Questi protocolli sono considerati mediamente più efficienti dei sistemi proof of stake ma presentano dei limiti e degli svantaggi ben evidenti. Innanzitutto con la proof of authority la rete assume un aspetto meno decentralizzato, in quanto pochi attori hanno la possiblità di influenzare l'intero sistema. Per questo motivo PoA è prevalentemente usato per le blockchain private. Altri punti di debolezza sono poi la difficoltà di trovare degli standard precisi per poter accettare un utente nel ristretto insieme di validatori e la complessità nel trovare maniere eque per scegliere il validatore di un blocco all'interno dei candidati. Un'altra critica frequente a questo protocollo è che l'identità pubblica dei validatori potrebbe portare più facilmente a manipolazioni di parti terze o a tentativi di corruzione.

Capitolo 2

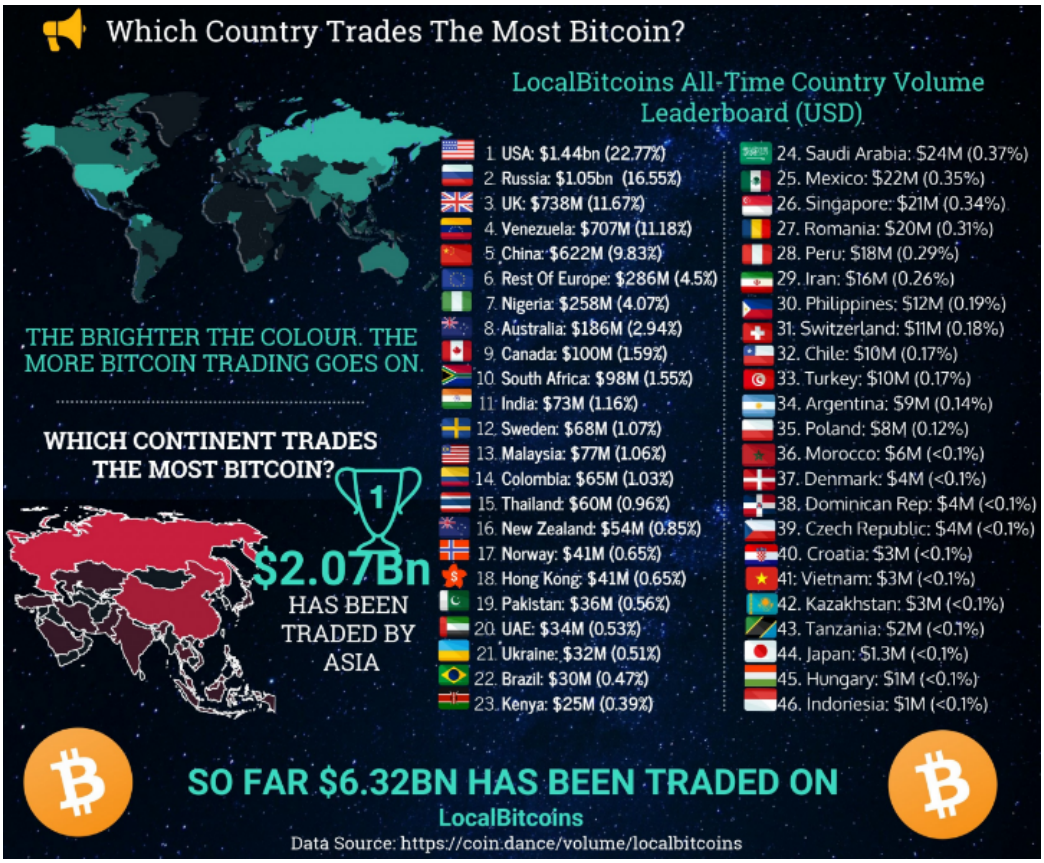
Bitcoin

Bitcoin è una criptovaluta e un sistema di pagamento digitale ideato nel gennaio 2009 da un inventore anonimo, noto con lo pseudonimo di Satoshi Nakamoto[17]. Generalmente quando viene scritto con la lettera maiuscola ci si riferisce al sistema vero e proprio mentre quando è scritto con la minuscola ci si riferisce alle singole monete digitali.
Bitcoin è un progetto open source scritto nel linguaggio C++ e utilizza come registro di tutte le transazioni la tecnologia blockchain, che ne permette un uso decentralizzato e fa sì che i dati siano distribuiti. Avendo la rete Bitcoin una strutturazione peer-to-peer non c'è la presenza di un ente centrale e ciò rende impossibile a qualunque autorità, governativa o meno, di bloccare i trasferimenti o di sequestrare le criptovalute, attività che sarebbero possibili solo con la conoscenza delle chiavi private degli utenti.
In Bitcoin gli utenti sono anonimi e si palesano al sistema tramite un indirizzo derivato dalla chiave pubblica che è composto dai 26 ai 35 caratteri alfanumerici.
Nel caso un utente smarrisca la chiave privata non sarà più in grado di accedere in alcun modo all'account e di recuperare le criptovalute possedute.
L'anonimità di Bitcoin non è esente da critiche, in quanto questa caratteristica può essere sfruttata per effettuare pagamenti per servizi illegali. Per questo e per altri motivi di natura economica alcuni stati hanno deciso di

bandirlo parzialmente o totalmente. Bitcoin è stato dichiarato illegale in Bolivia (2014), Bangladesh (2014), kyrgyzstan (2014), Marocco (2017), Nepal (2017) ed Ecuador (2015).



Per quanto riguarda il suo utilizzo nel mondo la maggior parte delle transazioni di Bitcoin vengono eseguite negli Stati Uniti, in Russia, in Cina, nel Regno Unito e in Venezuela. E proprio in Venezuela l'utilizzo di Bitcoin è particolarmente significativo visto che la recente brusca svalutazione del bolívar venezuelano ha portato molte persone a ritenere Bitcoin o altre criptovalute più affidabili della valuta locale. Inoltre è curioso notare come il Marocco, nonostante ufficialmente le criptovalute siano state dichiarate illegali nel paese, rientri al trentaseiesimo posto nella classifica per nazioni riguardo al giro di affari su Bitcoin. Questo va ulteriormente a dimostrazione di quanto sia difficile per i governi dei singoli stati controllare l'uso di questa tecnologia.



2.1 Transazioni

Una transazione è un trasferimento di bitcoin che viene propagato nella rete con lo scopo di essere inserito in un blocco. Quando un utente vuole effettuare un pagamento inserisce l'indirizzo del destinatario, la quantità di denaro da inviare e l'ammontare della commissione da pagare al minatore (fee). Il ricevente può vedere immediatamente dal suo software che la transazione è avvenuta ma non può essere subito sicuro che sia andata a buon fine. Solitamente entro circa 10 minuti la transazione sarà inserita in un blocco ma non è detto che questo farà parte della catena più lunga. Il ricevente può essere sicuro dell'avvenuto pagamento dopo circa un'ora, ovvero dopo

che sono stati minati teoricamente 6 blocchi.
Da un punto di vista tecnico la transazione è costituita da almeno un input e almeno un output. Per input si intende una precedente transazione in cui è stato inviato del denaro digitale al mittente del pagamento. Se questo denaro non è stato mai speso si parla di transazioni UTXO (Unspent Transaction Output), ovvero di crediti mai spesi che possono essere usati come input per futuri pagamenti. Per output si intendono invece i destinatari del trasferimento di monete digitali.
Si immagini il caso in cui il mittente usa come input due transazioni UTXO ciascuna di 0.1 bitcoin e vuole pagare per un servizio 0.15 bitcoin, inserendo 0.01 di fee. In questo caso il mittente invia a partire dagli input 0.15 bitcoin al destinario, 0.01 bictoin come ricompensa al minatore e 0.04 bitcoin a se stesso, in modo da non sprecare la parte del credito non utilizzata. La transazione da 0.04 bitcoin sarà una nuova transazione UTXO a credito del mittente, mentre gli input usati in precedenza non saranno più validi.

2.1.1 Script, un linguaggio per transazioni

Tramite Bitcoin si possono effettuare diversi tipi di transazioni. Innanzitutto bisogna specificare che gli indirizzi Bitcoin non sono delle chiave pubbliche, ma sono delle stringhe ottenute applicando una funzione di hash su tale chiave. Le transazioni avvengono tramite del codice scritto in un linguaggio stack-based non Touring completo chiamato **Script**, che è essenzialmente una sequenza di istruzioni che descrive i dettagli del trasferimento di denaro, assicurandosi della correttezza dell'operazione. Il linguaggio procede con l'esecuzione da sinistra verso destra, eseguendo le varie operazioni di push e pop. Alcuni dei comandi del linguaggio servono per effettuare operazioni logiche e aritmetiche sullo stack o per gestire il controllo di flusso, altri invece per gestire aspetti crittografici.

Ecco alcuni esempi:

Comando	Opcode	Descrizione
OP_EQUAL	0x87	Restituisce 1 se i due valori in cima allo stack sono uguali, 0 altrimenti
OP_IF	0x63	Se il valore in cima allo stack è *false* vengono eseguite le istruzioni, poi viene rimosso il valore in cima allo stack
OP_DUP	0x76	Duplica la cima dello stack
OP_SHA256	0xA8	Viene eseguita SHA256 sull'input
OP_HASH160	0xA9	Vengono eseguite sull'input prima SHA256 e poi RIPEMD-160
OP_HASH256	0xAA	Viene eseguito du volte SHA256 sull'input
OP_CHECKSIG	0xAC	Verifica che la firma per una transazione in input sia valida. In caso positivo ritorna 1, 0 altrimenti.
OP_VERIFY	0x69	Segna la transazione come non valida se il valore in cima allo stack non è true, poi rimuove la cima dello stack
OP_EQUAL VERIFY	0x88	Viene eseguita prima OP_EQUAL e poi OP_VERIFY

Il mittente della transazione in particolare dovrà fornire:

- una chiave pubblica che, quando sottoposta alla funzione di hash, ritorni l'indirizzo del mittente.
- una firma digitale che provi il possesso della chiave privata corrispondente alla chiave pubblica fornita.

In questo modo il mittente dimostra di essere in possesso degli input utilizzati per la transazione.
Script rende la modalità di pagamento più flessibile, facendo sì che i trasferimenti di denaro possano adattarsi a varie situazioni. Ad esempio in certi contesti potrebbero essere richieste due o più chiavi per autorizzare una transazione, oppure potrebbe non essere richiesta nessuna chiave.

2.1.2 Transazioni Pay-to-PubKey-Hash

La tipologia più comune di transazioni è chiamata Pay-to-PubKey-Hash (P2PKH) e si compone nella seguente maniera:

scriptPubKey:

OP_DUP OP_HASH160 $<$ *pubKeyHash* $>$ **OP_EQUALVERIFY OP_CHECKSIG**

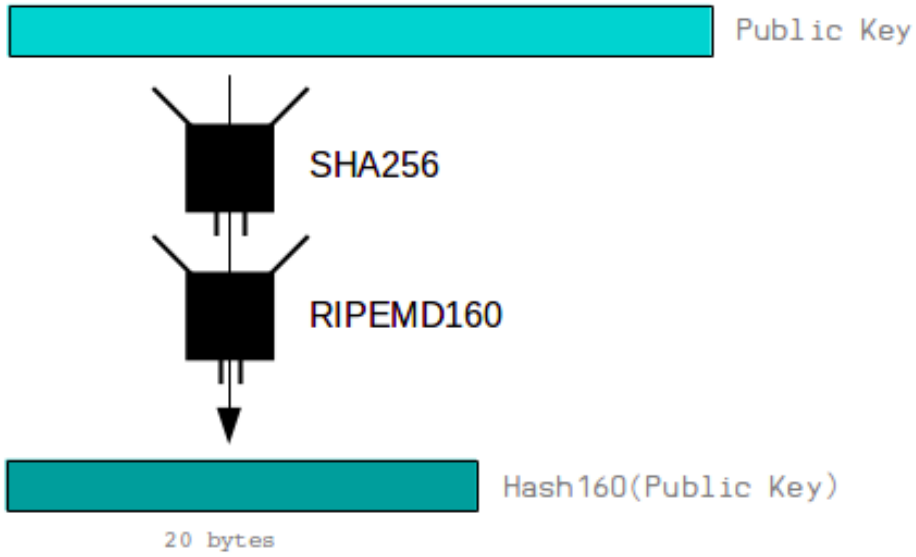
scriptSig:

$<$ *signature* $><$ *publicKey* $>$

In pratica quando lo script viene eseguito:

- La chiave pubblica viene duplicata (OP_DUP) e convertita in un Hash160 (OP_HASH160). L'Hash160 è la versione della chiave pubblica che vie-

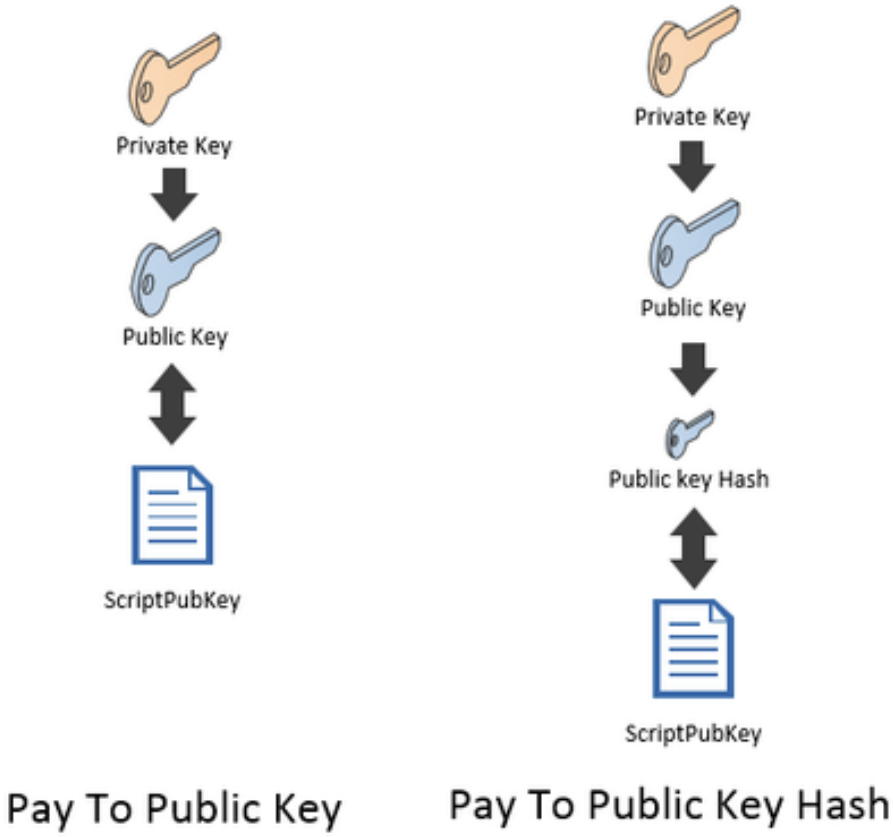
ne consegnata ai destinatari dei pagamenti, in modo da uffrire uno strato ulteriore di sicurezza. Per ottenere l'Hash160 si applicano in sequenza a partire dalla chiave pubblica le funzioni di hash SHA256 e RIPEMD10.



- Il valore dell'hash ottenuto viene comparato con l'hash della chiave pubblica presente in scriptPubKey per assicurarsi che corrispondano (OP_EQUALVERIFY).

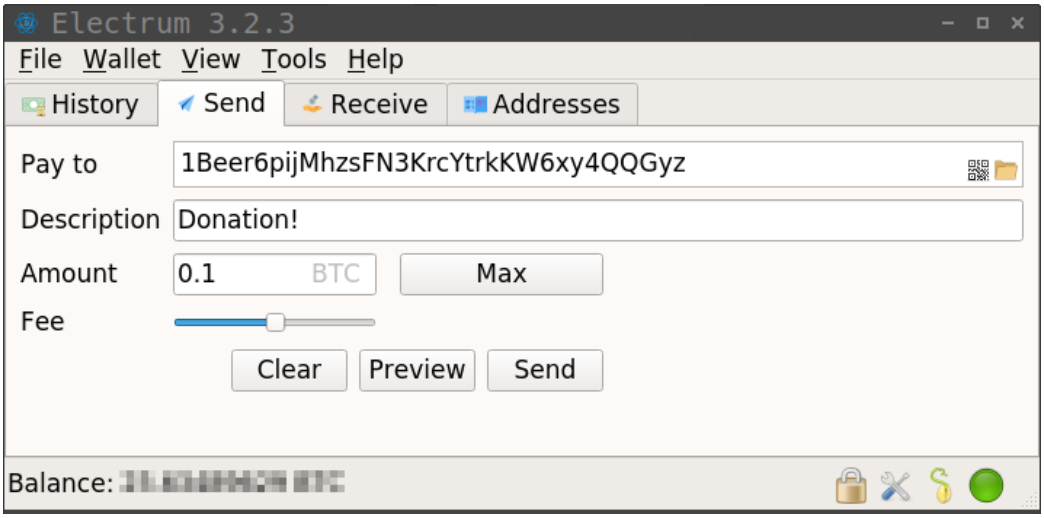
- Se corrispondono si procede a controllare la validità della firma digitale (OP_CHECKSIG).

Le transazioni P2PKH hanno soppiantato la versione usata precedentemente e considerata meno sicura chiamata P2PK (Pay to Public Key). P2PK è un algoritmo in cui viene divulgata la chiave pubblica (al posto dell'indirizzo come in P2PKH), che viene semplicemente confrontata con la firma digitale. Se viene verificata la validità della firma la transazione può essere inserita con successo.



I motivi che rendono preferibile usare P2PKH sono sia il fatto che l'hash160, che ha una dimensione di circa 20 byte, è facile da integrare in meccanismi di storage come i QR code, sia ragioni di sicurezza. Infatti P2PK potrebbe essere vulnerabile da parte di alcuni algoritmi per risolvere il problema del logaritmo discreto su curve ellettiche. Soprattuto in futuro, con il possibile avvento dei computer quantistici, P2PK potrebbe rivelarsi inadeguato. Pubblicando la chiave pubblica solo dopo che i soldi sono stati spesi c'è una garanzia maggiore di sicurezza.

Dal lato utente, in una reale applicazione verrà mostrata un'interfaccia simile:



Quando si mandano bitcoin a un indirizzo che comincia per 1, come nel caso dell'immagine sopra, significa che la transazione in questione è di tipo P2PKH.

2.1.3 Transazioni Pay to Script Hash

Pay to Script Hash (P2SH) è un tipo avanzato di transazione che permette il trasferimento di denaro verso l'hash di uno script valido. È uno strumento particolarmente utile per gestire le *multisignature*, ovvero quando è richiesta più di una firma per autorizzare una transazione. In questo caso non viene fornito l'hash della chiave pubblica ma l'hash dello script che serve a sbloccare gli input utilizzati per il pagamento (redeem script). Ad esempio m utenti potrebbero creare un indirizzo Bitcoin in comune, il quale necessita di $n <= m$ firme per acconsentire al trasferimento di denaro.

scriptPubKey:

OP_HASH160 $<$ *redeemScriptHash* $>$**OP_EQUAL**

scriptSig:

$< singatures >< publicKeys >< redeemScript >$

L'esecuzione di una transazione P2SH si divide in due parti. Nella prima viene controllata la correttezza del redeem script, assicurandosi della validità di tutti gli indirizzi e le firme digitali. Nella seconda parte lo script viene deserializzato, in modo da poter essere eseguito anche sulle singole componenti. Una delle caratteristiche interessanti collegate a P2SH è la possibilità di inserire dei vincoli nel trasferimento di denaro, per esempio facendo in modo che sia possibile spendere le criptovalute ricevute solo dopo una certa data.
A differenza delle transazioni P2PKH, in cui gli indirizzi cominciano tutti per 1, nelle transazioni P2SH gli indirizzi forniti iniziano con la cifra 3.

2.1.4 NULL DATA

Esiste inoltre un particolare tipo di script per le transazioni, chiamato NULL DATA. Come suggerisce il nome ogni output con uno script NULL DATA non è spendibile. Lo scopo dei NULL DATA è lo storage di dati nella blockchain. Ad esempio il mittente di una transazione potrebbe voler accompagnare un pagamento online con una descrizione testuale. Per evitare di sovraccaricare la blockchain con dati testuali però una transazione è considerata valida solo se è presente al massimo uno script NULL DATA.

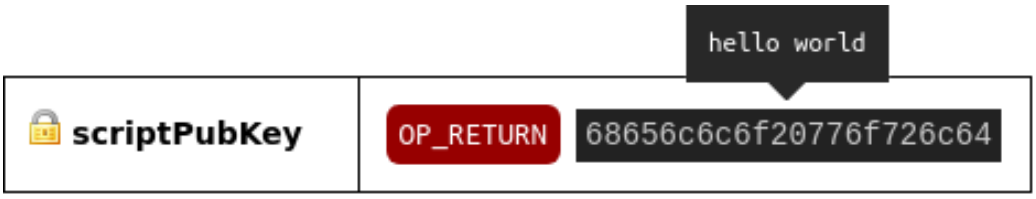


Figura 2.1: Esempio di script NULL DATA per scrivere "Hello World" nella blockchain

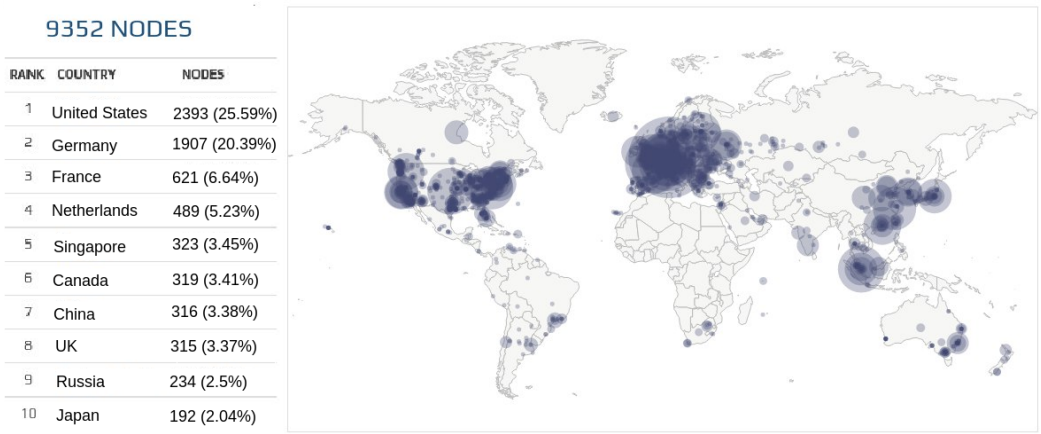
2.2 Topologia della rete

Il numero di utenti di Bitcoin è molto alto, si parla di circa 45 milioni di utenti complessivi[4]. Tra questi però ci sono alcuni nodi speciali, i full node, che svolgono diversi compiti indispensabili per la sicurezza e il mantenimento della rete, come quello di mantenere in locale l'intera blockchain e di assicurarsi della validità dei blocchi. In particolare devono essere rispettate alcune regole di consenso tra cui figurano:

- All'interno di un singolo blocco non è possibile spendere due volte lo stesso denaro.
- Blocchi e transazioni devono essere scritti nel formato corretto.
- Le transazioni devono avere delle firme corrette in riferimento ai bitcoin che sono stati spesi.
- La ricompensa per l'attività di mining deve essere quella stabilita dall'algoritmo condiviso.

Tra i full node ci sono alcuni nodi, i listener, che sono ancora più indispensabili al funzionamento del sistema perché oltre a svolgere le stesse funzioni degli altri full node si rendono visibili all'intera rete, fornendo le connessioni ai nuovi utenti che si uniscono per la prima volta al sistema.
I listener sono pubblici e quindi facilmente tracciabili e sono sparsi in diverse zone del pianeta con una particolare concentrazione negli USA, in Europa e nella parte occidentale della Cina.

Si stima invece che il totale dei full node superi le 100.000 unità.



9352 NODES

RANK	COUNTRY	NODES
1	United States	2393 (25.59%)
2	Germany	1907 (20.39%)
3	France	621 (6.64%)
4	Netherlands	489 (5.23%)
5	Singapore	323 (3.45%)
6	Canada	319 (3.41%)
7	China	316 (3.38%)
8	UK	315 (3.37%)
9	Russia	234 (2.5%)
10	Japan	192 (2.04%)

Figura 2.2: distribuzione dei nodi listener [5]

La topologia completa della rete di Bitcoin non è nota, anche se un attaccante potrebbe ottenere diverse informazioni a riguardo tramite l'uso di tecniche che analizzano le tempistiche di ricezione dei messaggi. Tramite modelli complessi che tengono conto delle latenze della rete e degli elaboratori e della distribuzione dei "degree" dei nodi potrebbe essere possibile ricostruire approssimativamente la topologia della rete [1] e utilizzare queste informazioni per rompere l'anonimità o per progettare in modo più mirato un denial of service contro un nodo specifico. Questa conoscenza potrebbe agevolare attacchi che puntano ad esempio a isolare un nodo dalla rete. Tuttavia, con semplici precauzioni come quella di inserire volontariamente alcuni ritardi casuali nella propagazione dei messaggi, gli attacchi sull'analisi delle tempistiche vengono resi impossibili, seppur al costo di un leggero rallentamento del sistema.

2.3 Blocchi

In Bitcoin i blocchi si possono dividere in tre categorie:

- main chain block, ovvero i blocchi inseriti con successo nella catena principale della blockchain
- stale block, ovvero i blocchi che sono stati minati con successo ma che non sono stati inseriti nella catena principale, probabilmente perché un altro blocco nella stessa posizione è stato esteso prima.
- orphan block, ovvero blocchi il cui blocco precedente non è presente nella blockchain locale e quindi non possono essere validati dalla rete.

La strutturazione di un blocco di Bitcoin si può schematizzare nella seguente maniera:

Field	Description	Size
Magic no	value always 0xD9B4BEF9	4 bytes
Blocksize	number of bytes following up to end of block	4 bytes
Blockheader	Consists of 6 items	80 bytes
Transaction counter	positive integer	1 - 9 bytes
Transactions	the (non empty) list of transactions	<Transaction counter>-many transactions

Field	Purpose	Updated when...	Size (Bytes)
Version	Block version number	You upgrade the software and it specifies a new version	4
hashPrevBlock	256-bit hash of the previous block header	A new block comes in	32
hashMerkleRoot	256-bit hash based on all of the transactions in the block	A transaction is accepted	32
Time	Current block timestamp as seconds since 1970-01-01T00:00 UTC	Every few seconds	4
Bits	Current target in compact format	The difficulty is adjusted	4
Nonce	32-bit number (starts at 0)	A hash is tried (increments)	4

Essenzialmente il blocco contiene del codice di controllo, uno header e la lista di transazioni organizzate come un Merkle Tree. Nello header vengono indicate la versione precisa del procollo utilizzato, l'hash del blocco precedente (grazie a cui è possibile ricostruire la catena), il timestamp, l'hash del Merkle root e altri due campi, Bits e Nonce, che servono per le operazioni di mining. Bits indica la difficoltà di minare un blocco e Nonce è l'insieme

dei valori che il miner prova a indovinare per convalidare il blocco. Essendo un campo di 4 byte ci sono 2^{32} possibili combinazioni da provare. Nel caso nessuna di queste produca un hash valido è possibile proseguire le operazioni di mining cambiando il timestamp del blocco o cambiando l'ordine in cui le transazioni vengono disposte nel Merkle Tree. Infatti anche solo un piccolo cambiamento sul blocco produce un grande cambiamento sull'hash che viene calcolato, quindi anche il solo aggiornamento del timestamp offre nuovi miliardi di nuovi tentativi per effettuare l'attività di mining con successo.

2.4 Attacchi alla sicurezza

La natura decentralizzata di Bitcoin unita al meccanismo di consenso della proof of work rendono impossibile per un attaccante falsificare le transazioni di denaro già inserite all'interno di blocchi e propagate all'intera rete. È però possibile effettuare altri tipi di comportamenti malevoli, come ad esempio il **double spending**. Un utente potrebbe infatti decidere di provare a spendere il denaro proveniente dalla medesima transazione UTXO due volte. I meccanismi di controllo delle inconsistenze non permetteranno di inserire entrambe le transazioni nella catena principale della blockchain, per cui una di queste verrà scartata. Il double spending però non mette a rischio la sicurezza dell'intero sistema ed è risolvibile aspettando un periodo di tempo opportuno prima di erogare il servizio richiesto, in modo da essere sicuri che il pagamento sia effettivamente andato a buon fine. Questo certamente limita l'utilizzo di Bitcoin, in quanto non lo rende adatto o completamente sicuro[12] ad alcune situazioni in cui non è possibile o non è pratico aspettare una tale quantità di tempo.
Cosa invece potrebbe mettere potenzialmente in crisi il funzionamento dell'intero sistema è un **attacco del 51 %**, ovvero una situazione in cui un utente o un gruppo di utenti controlla più del 50% della potenza computazionale dell'insieme dei minatori. Gli attaccanti sarebbero in grado di impedire alle transazioni a loro sgradite di ricevere conferma, rendendo impossibile per

alcuni utenti o volendo anche per l'intero sistema di eseguire un qualsiasi tipo di pagamento. Chi fosse in controllo di tale potenza di calcolo di hash potrebbe inoltre effettuare double spending sapendo con relativa sicurezza di poter inserire nella catena principale della blockchain la transazione fasulla, facendo invece scartare l'altra.

Un altro attacco che potrebbe essere portato a termine da un utente malintenzionato è quello del **selfish mining**[28], ovvero una situazione in cui l'attaccante non pubblica immediatamente il blocco che è riuscito a minare, ma prova direttamente a minarne uno successivo, con la speranza di raggiungere un vantaggio di due unità rispetto alla catena pubblica e quindi di rendere inutile tutta l'attività di mining degli altri nodi della rete. Infatti visto che in Bitcoin i nodi si accordano per considerare come catena principale quella di lunghezza maggiore, se l'utente malintenzionato mantiene in privato una serie di blocchi maggiore in lunghezza alla catena principale della rete, quando deciderà di divulgare i blocchi questi faranno parte della catena principale, e tutto lo sforzo computazionale impiegato per minare gli altri blocchi viene reso vano. Questo comportamento, soprattutto nel caso l'hash-rate dell'attaccante non sia così elevato, può però essere controproducente in quanto il blocco minato segretamente potrebbe non raggiungere mai la rete e quindi non si otterrebbe nessuna ricompensa per il lavoro computazionale svolto.

Un altro potenziale problema per la rete è il cosiddetto **Sybil attack**, un attacco in cui un utente prova ad assumere il controllo dell'intera rete creando un'alta quantità di nodi fasulli nel sistema che in realtà sono controllati dall'attaccante. Creando un numero sufficiente di identità false l'aggressore potrebbe riuscire a mettere in minoranza i nodi onesti e rifiutarsi di trasmettere loro una parte o anche la totalità delle transazioni e dei blocchi ricevuti, isolandoli dalla rete. Un utente potrebbe sfruttare questo attacco in diversi modi ad esempio:

- rifiutandosi di trasmettere qualsiasi messaggio verso un nodo o proveniente da un nodo, isolandolo completamente dalla rete. In questo caso si parla di un attacco di DoS filtering.
- trasmettendo solo i suoi blocchi in modo da assicurarsi che i blocchi minati dall'attaccante finiscano nella catena principale.
- filtrando le transazioni sgradite, in modo da facilitare l'attuamento di un double spending.
- cercando di rompere l'anonimità degli utenti.

È rilevante sottolineare che un attacco assume un significato differente a seconda delle tecniche di validazione dei blocchi utilizzate. Ad esempio l'attacco del 51% assumerà un significato diverso a seconda se sia stato utilizato un protocollo proof of work o proof of stake. Nel PoW infatti l'attaccante per avere successo deve avere a disposizione più della metà della potenza computazionale utilizzata dai miner, nel PoS invece sarà necessario possedere più della metà del denaro depositato come cauzione.
Il Selfish mining invece è un tipo di comportamento malevolo che solitamente non è attuabile in protocolli diversi dal proof of work, in quanto si basa sulla possibilità di minare blocchi concorrentemente e di non divulgare subito i blocchi validi.
Esistono anche attacchi che sfruttano la semantica del proof of stake, in particolare il cosiddetto **nothing at stake**[27]. L'attacco si basa sul fatto che, in caso di biforcazione, è di interesse del validatore inserire le nuove transazioni su entrambi i fork. Questo perché, a differenza del PoW, creare un blocco è un'attività a costo zero e il validatore, che vuole assicurarsi di ricevere una ricompensa per l'aggiunta del blocco a prescindere da quale sarà la catena più lunga, è incentivato a proseguire tutte le biforcazioni. Nel proof of work questo non è ovviamente possibile: il miner dovrà scegliere quale catena proseguire e sperare di essere più veloce degli avversari.
Uno scenario realistico di Nothing at Stake potrebbe essere quello in cui un utente malintezionato vuole effetuare double spending e quindi quando viene

selezionato per validare un blocco crea una biforcazione, inserendo in una catena la transazione reale e in un'altra quella fasulla. Dopodiché l'attaccante attenderà di essere nuovamente selezionato come validatore, confidando sul fatto che entrambe le catene verranno mantenute attive. Quando sarà di nuovo il suo turno sceglierà di proseguire solo la catena in cui è presente la transazione fasulla, portando così a termine l'attacco. Per arginare questo problema alcuni protocolli PoS come Casper (il meccanismo di validazione dei blocchi che verrà usato da Ethereum quando ci sarà la transizione da PoW a PoS) penalizzano economicamente i validatori che aggiungono blocchi su più catene.

2.5 Aspetti economici

Uno degli aspetti rivoluzionari di Bitcoin e delle altre criptovalute è il fatto che queste monete siano sottratte alle fasi di emissione e controllo, compito solitamente della banca centrale.
A differenza della valute tradizionali Bitcoin non fa riferimento a nessun ente governativo che ne regoli valore, né ci sono riserve auree o beni di altro tipo a fungere da garanzia. Il valore della criptovaluta nel mercato è determinato esclusivamente dalla legge della domanda e dell'offerta e potrebbe improvvisamente deprezzarsi nel caso, a causa di qualche evento imprevisto, dovesse rompersi la fiducia che gli usufruitori della criptovaluta hanno nei confronti del sistema.
In Bitcoin la quantità di moneta virtuale presente nel sistema è limitata a priori ed è conoscibile da qualsiasi utente. Tale quantità aumenta nel tempo perché vengono immessi in circolazione nuovi bitcoin come ricompensa per l'attività di mining. Questa ricompensa, che inizialmente era di 50 bitcoin a blocco, si dimezza ogni 210000 blocchi minati.
La quantità di bitcoin totali nel sistema tende a 21 milioni e questa soglia verrà raggiunta nel 2140. Dopo questa data, ovvero dopo che sarà stato minato il blocco numero 6930000, se il sistema sarà ancora in uso i minatori

potranno guadagnare solo tramite le fee inserite nelle transazioni.

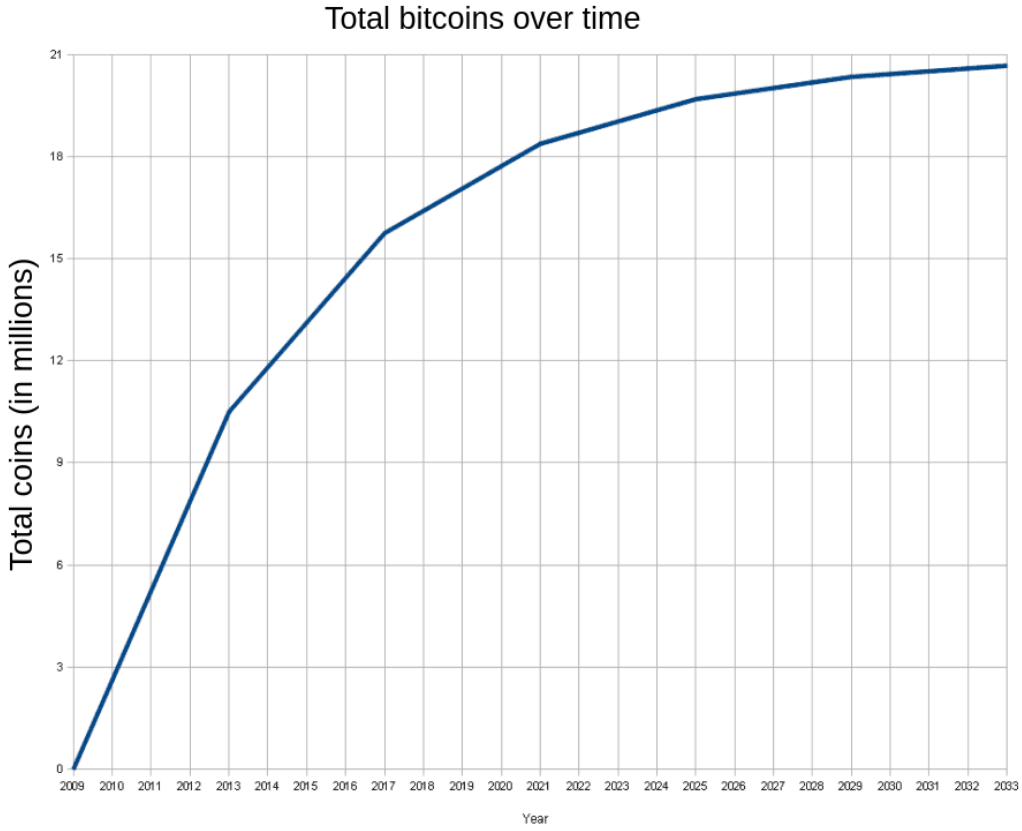


Figura 2.3: Numero di bitcoin totali nel sistema dal 2009 al 2033 [5]

Per quanto riguarda il valore monetario, il cambio bitcoin-dollaro è stato sempre molto variabile nel tempo, con cicli composti da rapide impennate e improvvisi deprezzamenti.
Pur con le continue oscillazioni il valore di un bitcoin è stabilmente sopra i 1000 dollari da febbraio 2017, con un picco di 19783.21 dollari del 17 dicembre 2017 [29].
Originariamente il valore monetario era molto più basso. Il primo pagamento con bitcoin infatti fu infatti un acquisto di pizze presso la catena ristoratrice statunitense Papa John's, che vennero consegnate per la somma di 10000

bitcoin, ammontare che oggi supererebbe di gran lunga il milione di euro.

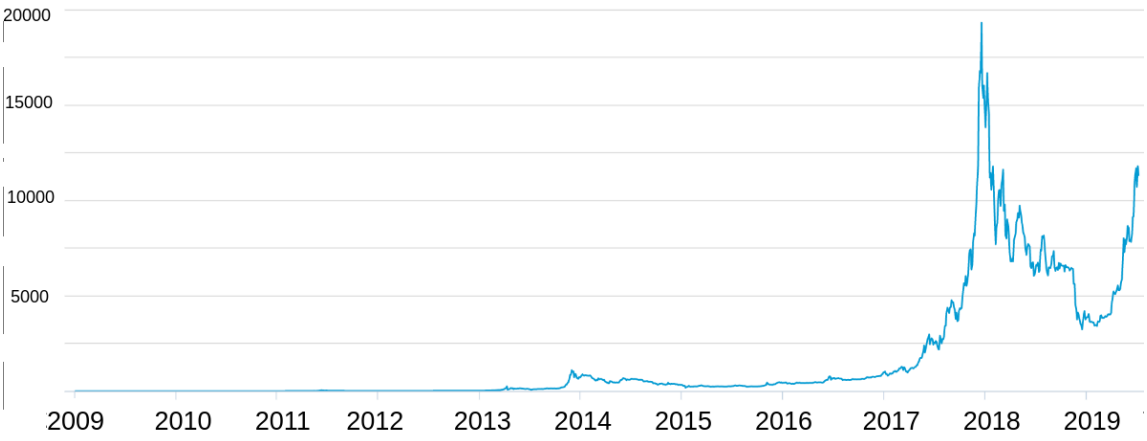


Figura 2.4: Cambio bitcoin-dollaro nel corso degli anni [5]

Si pensa che uno dei motivi principali dell'oscillazione del valore dei bitcoin sia la speculazione economica di alcuni degli stakeholder, oltre allo scetticismo derivante da prese di posizione avverse da parte di alcuni singoli stati contro le criptovalute. È ipotizzabile dunque che col crescere dell'utenza del sistema e con una legislazione internazionale più chiara si potrà avere una situazione finanziariamente più stabile.

Capitolo 3

Simulazione di sistemi

Con il termine simulazione si intende riprodurre il comportamento di un sistema tramite l'uso di un modello progettato ad hoc. In alcuni casi il sistema simulato è già esistente e funzionante, in altri invece deve essere ancora progettato e implementato e il modello potrà fornire informazioni utili riguardo alle performance e alla consistenza del progetto.
I motivi per effettuare simulazioni possono essere di vario tipo, come la riduzione dei costi sui test, il pericolo di testare direttamente un sistema reale (si pensi ad esempio a un'applicazione medica da cui dipende la vita di un paziente) o la necessità di testare il funzionamento delle singole componenti individuali all'interno di un progetto di elevata complessità.
Descrete Event Simulator (DES) è un paradigma di simulazione [10] che spesso è in grado di combinare verosimiglianza della rappresentazione e facilità d'uso. Nei DES il tempo è suddiviso in un certo numero di intervalli discreti e gli eventi accadono in uno di questi passi della simulazione, in ordine cronologico. Si può dire quindi che un DES è costituito da:

- un insieme di variabili di stato associate ai vari agenti del sistema
- una lista di eventi, ovvero un elenco di eventi futuri che quando verranno eseguiti modificheranno gli stati del sistema.
- un clock globale, che scandirà i vari step discreti della simulazione.

Idealmente gli eventi sono etichettati con un intero che rappresenta il timestep della simulazione in cui sono stati eseguiti, rendendo possibile una ricostruzione cronologica di ciò che è accaduto.
Spesso poi è fondamentale avere accesso a una fonte randomica che scandisce la generazione di eventi. Può essere importante fare diversi test con diversi generatori di numeri pseudocasuali per assicurarsi della robustezza dell'applicazione, che non deve fallire anche nel caso in cui si verifichino combinazioni di avvenimenti rari e/o particolari.
In una simulazione sequenziale una singola PEU (Physical Execution Unit) ha il compito di eseguire gli eventi programmati, occupandosi dell'intera gestione del modello e della sua evoluzione nel tempo. Questo approccio è sicuramente tra i più semplici da implementare ma comporta una serie di limiti di scalabilità, in quanto potrebbe essere richiesta una grande quantità di tempo per l'esecuzione oppure potrebbe non essere possibile rappresentare grandi quantità di dati.
Per superare i vincoli di scalabilità può essere opportuno usare un approccio PDES (Parallel Discrete Event Simulation), che parallelizza alcune attività tramite l'uso di più unità fisiche di esecuzione tra di loro interconnesse. In questo modo è possibile velocizzare l'esecuzione della simulazione al costo di un leggero aumento della complessità.

Per migliorare ulteriormente le performance è possibile seguire un approccio PADS (PArallel and Distributed Simulations) in cui la simulazione è eseguita su più di un processore. I vantaggi di questo approccio consistono nei miglioramenti sulla velocità di esecuzione, sulla scalabilità del modello e sull'interoperabilità del sistema. Inoltre sarà possibile integrare più simulatori preesistenti in uno solo modello di simulazione. Nei PADS, a differenza dei simulatori sequenziali, non c'è uno stato globale del modello. Ogni PEU infatti si incaricherà di gestirne solo una parte e le componenti del modello eseguite dalla singole unità di esecuzione sono dette LP (Logical Process). Quando i vari LP girano sullo stesso elaboratore e sono connessi tramite la

memoria condivisa si parla di simulazione parallela, invece quando i logical process sono situati in diverse macchine e collegati tramite un'infrastruttura di rete si parla di simulazione distribuita. La capacità di trasmissione della rete avrà un'influenza non trascurabile sulla velocità di esecuzione della simulazione. Aspetti fondamentali per l'uso di un'approccio PADS sono la sincronizzazione temporale tra i logical process e la distribuzione di task e dati tra i vari LP, partizione che deve essere fatta cercando di bilanciare il lavoro delle unità di esecuzione e allo stesso tempo di minimizzare il più possibile le comunicazioni.
Per garantire l'integrità della simulazione degli avvenimenti è necessario che l'ordine causale degli eventi non venga violato. In pratica se un evento A avviene prima di un evento B e se l'azione di A può influenzare l'esito di B, allora è necessario che A venga simulato prima di B. In un ambiente sequenziale è facile garantire che gli eventi vengano eseguiti in ordine non decrescente, in un ambiente distribuito la questione è più complicata a causa delle diverse performance delle macchine e dei delay per la comunicazione. È perciò necessario adottare un algoritmo di sincronizzazione. Ci sono diversi tipi di approccio:

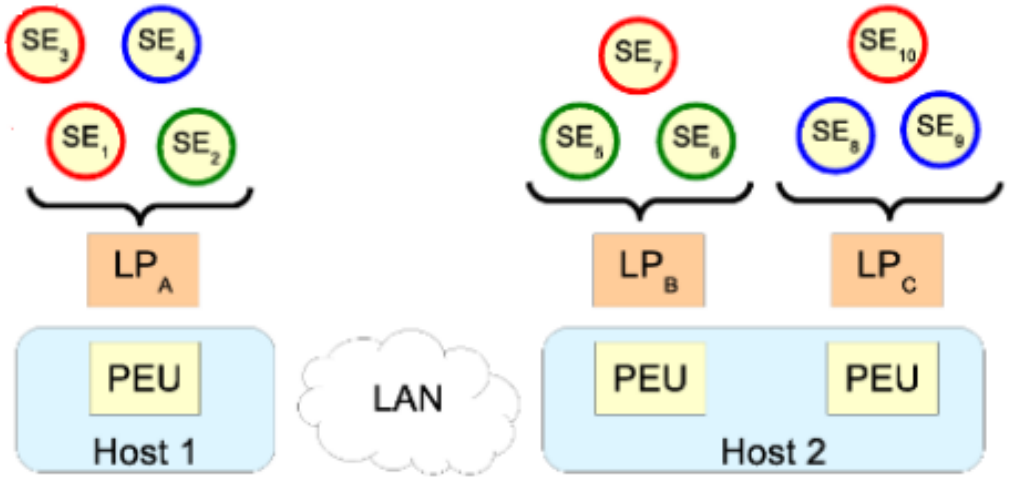
- time-stepped. Il tempo viene suddiviso in timestep di dimensione nota. In ogni passo discreto della simulazione il modello viene aggiornato e i logical process comunicano gli eventi eseguiti agli altri LP. Quando tutti i logical process hanno terminato l'esecuzione si sincronizzano e possono cominciare il passo successivo della simulazione.

- conservativo. Questo approccio mira a garantire che non vengano violati i vincoli di causalità. Prima di eseguire un evento viene controllato se questo è sicuro, ovvero se è impossibile che la sua esecuzione provochi errori di causalità. Se l'evento non è sicuro il logical process sospenderà temporaneamente l'esecuzione degli eventi fino a che non sarà nuovamente sicuro eseguirli.

- ottimistico. Gli eventi sono eseguiti in ordine di ricezione e c'è quindi

un'alta probabilità che venga violato l'ordine causale. È perciò è necessario conservare gli stati precedenti e annullare l'esecuzione di eventi quando vengono scoperte violazioni ai vincoli causali.

Il modello poi può essere suddiviso in piccole componenti chiamate simulated entity (SE). Ogni SE sarà assegnata a uno dei logical process, che svolgerà quindi la funzione di contenitore di SE. Le entità simulate possono anche cambiare LP di competenza, in questo caso si parlerà di migrazione.



Un tipo particolare di simulazione è chiamato Agent Based Modeling and Simulation (ABMS). Negli ABMS le entità simulate sono chiamate agenti e ogni agente rappresenta un qualche tipo di attore presente nella simulazione. Il modello specifica il compotamento degli agenti, che spesso è influenzato dalle informazioni che questi ricevono dall'ambiente. Perciò gli eventi generati nell'ambiente possono avere un impatto sugli stati degli agenti e l'intera simulazione si evolve basandosi sulle interazioni tra modello e agenti.

In un simulatore time-stepped a volte può essere utile utilizzare un approccio multilivello, per far sì che diversi passi della simulazione rappresentino una diversa quantità di tempo simulata. In alcuni modelli potrebbero infatti esserci sia frequenti avvenimenti significativi che accadono durante brevi in-

tervalli di tempo, sia intervalli temporali relativamente lunghi in cui accade poco o nulla di rilevante.
Un approccio multilivello è particolarmente adatto per modellare applicazioni che fanno uso della rete. Il tempo per propagare l'informazione tra i vari nodi è generalmente bassissimo e gli altri eventi simulati potrebbero essere collocati in un intervallo temporale molto più grande del tempo necessario per lo scambio di un messaggio. In questi casi potrebbe avere senso usare due tipi di step nella simulazione: uno per la propagazione dei messaggi e un altro per simulare gli altri eventi di diversa natura.

3.1 LUNES

LUNES (Large Unstructured NEtwork Simulator) è un programma che consente di effettuare simulazioni su reti complesse composte da un alto numero di nodi. Scritto nel linguaggio di programmazione C per massimizzarne l'efficienza, LUNES segue l'approccio PADS in modo da avere una strutturazione tale per cui sia possibile svolgere la simulazione anche parallelamente su più unità elaborative.
Per quanto riguarda la gestione dei passi di simulazione è stato invece utilizzato un approccio time-stepped, in cui ogni passo temporale rappresenta il tempo necessario per due nodi vicini di mandarsi un messaggio. Nel simulatore a ogni logical process viene assegnato un certo numero di simulated entity, che in questo caso specifico rappresentano un singolo nodo della rete.
LUNES è stato concepito modularmente, in quanto il programma è costituito essenzialmente da tre fasi che vengono eseguite separatamente:

- La creazione della topologia della rete, affidata alla libreria di C i-graph, con cui si possono costruire grafi con diverse caratteristiche e con un numero variabile di nodi e archi.

- La simulazione del protocollo, che durerà il numero prefissato di passi discreti.

- La valutazione delle performance, che si basa sui dati raccolti durante la simulazione.

LUNES per effettuare le simulazioni si appoggia al middleware ARTIS e al framework GAIA.
ARTIS (Advanced RTI System)[9] è un middleware adattivo, ovvero uno strumento che fornisce alle applicazioni servizi aggiuntivi rispetto a quelli offerti dal sistema operativo. Il middleware agisce facendo da intermediario tra il kernel e il software utilizzato dall'utente e adattandosi alla situazione corrente dello scenario d'uso. ARTIS in particolare gestisce gli aspetti di comunicazione tra i vari LP, supportando la simulazione in ambienti paralleli e distribuiti. È progettato per offrire un alto grado di scalabilità, supportando esecuzioni costituite da un alto numero di Physical Execution Unit (PEU).
In diversi simulatori come LUNES le performance sono fortemente influenzate dall'efficienza delle comunicazioni tra le varie simulated entity, per cui è fondamentale ottimizzare questi aspetti. Ad esempio i logical process che sono situati nella stessa unità fisica di esecuzione comunicano con la memoria condivisa, gli altri invece comunicano tramite la rete con i protocolli TCP/IP.
ARTIS si avvale di un SImulation MAnager (SIMA), che lo rende un software con un'architettura parzialmente centralizzata. I compiti del SIMA sono quelli di gestire l'inizializzazione e la terminazione della simulazione e quello di occuparsi, durante l'esecuzione, della coordinazione tra i logical process e delle varie attività di sincronizzazione.
Le API a disposizione per utilizzare il SIMA sono:

- *void SIMA_Initialize (int port, int numpeers, char *netfile)*, con cui il SIMA apprende quanti sono i nodi che partecipano alla simulazione, qual è la porta da utilizzare per le comunicazioni e qual è il file che contiene le configurazioni dei canali di comunicazione.

- *void SIMA_Finalize()*, con cui si chiude la comunicazione tra SIMA e LP e si rilasciano le risorse allocate nella fase di inizializzazione.

- *void SIMA_Barrier()*, con cui si crea un punto di sincronizzazione durante l'esecuzione tra il SIMA e tutti gli altri logical process.

GAIA (Generic Adaptive Interaction Architecture) usa il backend di ARTIS e gestisce gli aspetti di comunicazione tra le varie simulated entity.
Utilizzando il SIMA di ARTIS e GAIA non è necessario quindi implementare nuovi meccanismi di gestione delle comunicazioni. GAIA è un framework migration-based, in cui per migrazione si intende lo spostamento di una simulated entity da un logical process di competenza a un altro. Le primitive di comunicazione offerte sono:

- *void GAIA_SetLoadBalancing (int state)*, che consente di attivare o disattivare le funzionalità di load balancing. Nel caso si sia deciso di utilizzare il load balancing alcune delle simulated entity allocate nei logical process più lenti verranno spostate su LP più performanti.

- *void GAIA_Migrate(int id, String data, int size)*, utilizzata per effettuare le procedure di migrazione da un LP locale a uno remoto.

- *GAIA_Send()*, per inviare messaggi.

- *GAIA_Receive(max_len)*, per ricevere messaggi

- *GAIA_Initialize()*, per permettere a un logical process di entrare nella simulazione.

- *GAIA_Finalize()*, per permettere a un logical process di uscire dalla simulazione.

In LUNES è possibile implementare protocolli di gossip e valutarne le prestazioni. Un protocollo di gossip è un algoritmo per la diffusione di messaggi all'interno di una rete peer to peer. In un sistema distribuito, a differenza delle architetture client-server, non c'è un nodo o un gruppo di nodi master a cui rivolgersi per ottenere i dati, ma questi dovranno essere diffusi all'interno della rete con meccanismi di propagazione che coinvolgono i vari

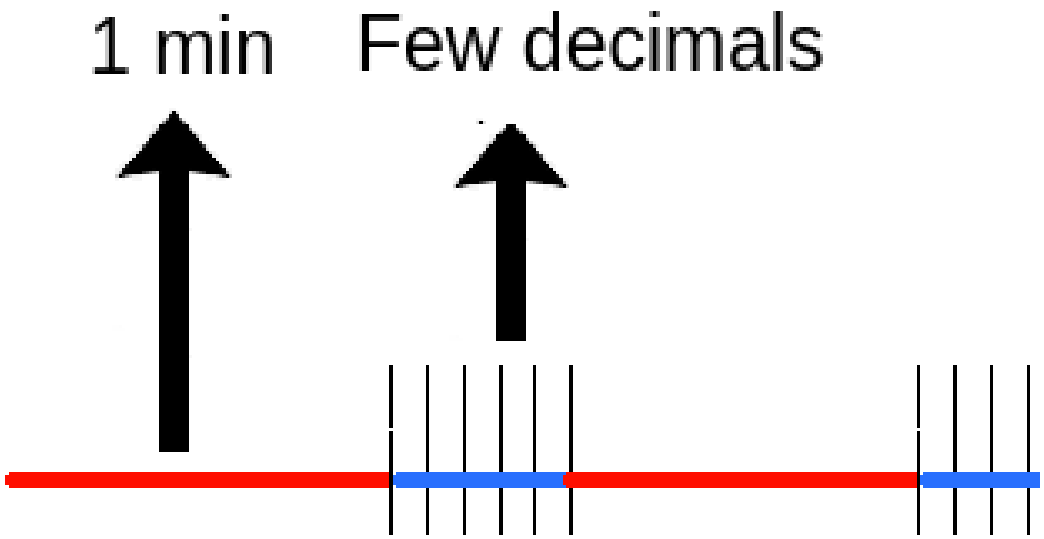
peer del sistema. Le preoccupazioni principali sono distribuire i dati a tutti i partecipanti e minimizzare il numero di messaggi necessari per ottenere una copertura completa. Una situazione da evitare è infatti quella in cui un messaggio, già ricevuto da tutti i componenti della rete, continua a essere inoltrato dai vari nodi. Per evitare queste situazioni, al di là delle specifiche tecniche utilizzate dai protocolli di gossip, è possibile creare una cache e/o attribuire un time to live (TTL) ai messaggi. La cache è un meccanismo che tiene traccia degli ultimi messaggi ricevuti. Quando si riceve un messaggio che è già presente nella cache, questo non verrà inoltrato ai nodi vicini. Il TTL invece impedisce che un messaggio venga inoltrato all'infinito. Impostato inizialmente con un valore, che di solito è appena superiore al diametro massimo della rete, il TTL decresce di un'unità ogni volta che il messaggio viene consegnato a un nodo. Quando questo valore raggiunge lo 0 il messaggio non verrà più inoltrato ai vicini. Queste due tecniche possono essere anche usate in combinazione.

3.2 LUNES-bitcoin

LUNES-bitcoin si basa su LUNES ma in aggiunta è presente del codice che mira a simulare il sistema Bitcoin, in particolare negli aspetti di creazione, gestione e propagazione dei blocchi. Sono presenti dunque strutture dati per rappresentare blocchi e transazioni.

A differenza di LUNES inoltre è stato usato un approccio multilivello per quello che riguarda la gestione dei timestep. Sono infatti presenti due tipi di step temporali, uno (propagation step) che rappresenta una durata di tempo brevissima, necessaria solo allo scambio di un messaggio tra due nodi vicini e un altro (mining step) della durata di un minuto circa in cui i nodi, a seconda dell'hashrate a loro assegnato, hanno una certa probabilità di minare un blocco.
Col nuovo approccio possono comunque crearsi conflitti sulla catena, in quan-

to è possibile che nello stesso mining step due nodi minino un blocco nella medesima posizione. In questo caso si genera una race condition. Al mining step successivo però tutti i nodi, salvo interferenze dovute ad attori malintenzionati, avranno ricevuto i blocchi minati precedentemente. Durante i vari propagation step infatti i messaggi contenenti i blocchi minati vengono propagati su tutta la rete.



Il programma è costituito dai seguenti file:

In msg_*definition.h* vengono definiti i messaggi che possono essere inviati tra i nodi durante la simulazione. In particolare ci sono 5 tipi di messaggi:

- TransMsg, per diffondere una transazione
- BlockMsg, per diffondere un blocco
- AskMsg, per richiedere dei blocchi che ancora non si trovano nella blockchain locale del mittente
- LinkMsg, per la costruzione della rete
- MigrMsg, per informare un LP di una migrazione

In tutte le tipologie di messaggi è presente il campo *type*, che serve a capire di che tipo di messaggio si tratta.
In *lunes.c* si trovano le funzioni che servono a definire il comportamento da tenere in risposta alla ricezione di un messaggio oltre ai metodi che incapsulano le procedure di invio di messaggi relativi a blocchi e transazioni sulla blockchain.

In *user_event_handlers.c* si trovano funzioni per creare e gestire l'invio di messaggi, oltre che a procedure di supporto per queste funzionalità e metodi per raccogliere statistiche.
bitcoin.c è il file che viene chiamato direttamente dallo script che esegue la simulazione. Qui dentro viene caricato il grafo che rappresenta la rete, vengono caricate le simulated entity, inizializzati i vari campi di ogni nodo e gestiti i vari step della simulazione.

Capitolo 4

Valutazione delle performance dei protocolli di gossip

L'obiettivo di questi esperimenti è quello di testare le performance di 3 diversi protocolli di gossip, ovvero:

- Fixed Probability. Consiste nell'inviare un messaggio a un vicino nel caso un numero generato casualmente superi un certa soglia. Si ripete poi l'operazione per ogni vicino del nodo.
- Probabilistic Broadcast. Consiste nell'inviare il messaggio a tutti i vicini solo se un numero generato casualmente supera una certa soglia, in caso contrario non verrà inviato nessun messaggio.
- Dandelion, il protocollo di gossip che viene usato dalla rete Bitcoin per la diffusione delle transazioni.

Dandelion presenta due fasi:

- Stem phase. In questa fase il messaggio viene inoltrato a un singolo vicino, scelto casualmente tra la lista di vicini del nodo.
- Fluff phase. In questa fase il messaggio viene inoltrato in broadcast, ovvero tutti i vicini del nodo riceveranno il messaggio.

La motivazione per l'uso di questo protocollo di gossip è legata al mantenimento dell'anonimità del mittente della transazione. Infatti, anche nel caso in cui un attaccante riuscisse a capire la locazione da cui ha avuto origine la fluff phase, l'identificazione del nodo che ha creato la transazione e da cui quindi ha avuto origine la stem phase risulterebbe molto difficile.

Probabilistic Broadcast e Fixed Probability sono concettualmente più simili in quanto il parametro *prob* che ricevono in input ha il medesimo significato. Infatti *prob* indica la probabilità che un messaggio venga inoltrato a un singolo nodo (nel caso di fixed probaility) o a tutti gli altri nodi vicini (nel caso di Probabilistic Broadcast). Per Dandelion invece il parametro che viene preso in considerazione indica il numero di passi di fluff phase, in cui il messaggio sarà propagato in broadcast. I passi di stem phase corrispondono ovviamente al TTL (time to live) meno il numero di step della fluff phase.
I test sono stati effettuati su una rete di 500 nodi e con un numero variabile di archi. Il TTL è stato sempre impostato a 12 e la dimensione della cache a 256 messaggi. Inizialmente il TTL di default era 16. Dopo i primi test si è cercato di capire le conseguenze di abbassare il time to live da 16 a 12 (ovvero a una misura appena superiore a 10, il diametro massimo), e i risultati sono stati praticamente identici, con solo differenze trascurabili dovute agli eventi probabilistici. Questo è sintomo del corretto funzionamento della cache, che impedisce l'inoltro di messaggi già ricevuti. Si è scelto dunque di proseguire impostando il TTL a 12 per evitare computazioni non necessarie.

Per valutare l'efficienza dei protocolli sono state prese in considerazione tre diverse misurazioni:

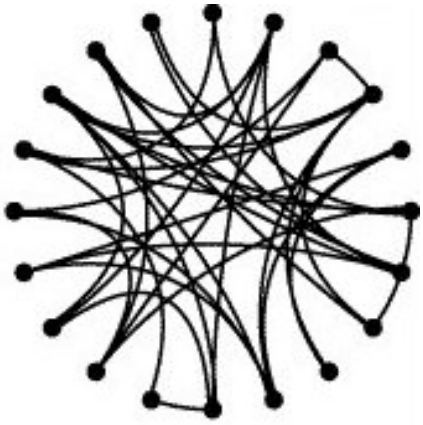
- Coverage, ovvero la percentuale di nodi che ricevono un messaggio che è stato mandato attraverso le rete.
- Delay, che rappresenta il numero medio di hop che il messaggio compie per raggiungere un nodo. Infatti quando un messaggio viene ricevuto per la prima volta da un nodo viene registrato il numero di hop

impiegati per raggiungerlo a partire dal nodo creatore. Il delay è quindi calcolato come il numero medio di hop impiegati da un messaggio per raggiungere un nodo, facendo la media su tutti i messaggi inviati durante l'esecuzione del test.

- Overhead, definito come il rapporto tra il numero di messaggi consegnati e il limite inferiore della rete, cioè il teorico numero minimo di messaggi nel grafo per ottenere la copertura completa.

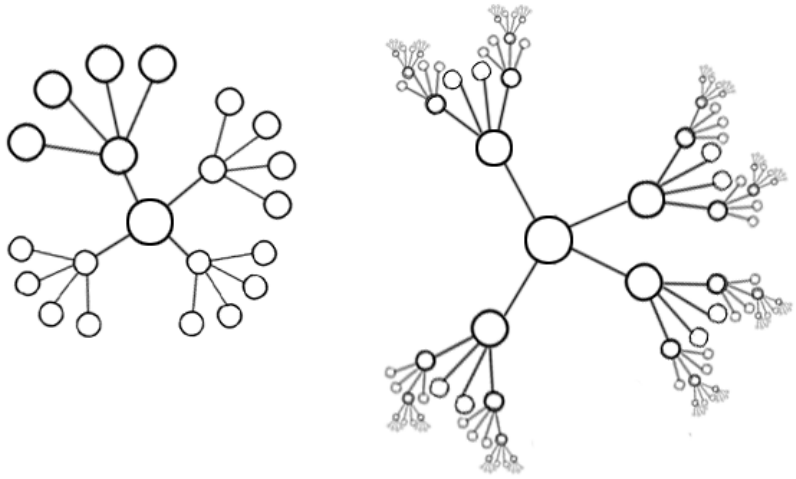
I test sono stati fatti su 4 tipi differenti di grafi:

- randomici, grafi in cui i collegamenti tra i nodi vengono generati completamente in modo casuale.
  Per creare grafi randomici si usa solitamente il modello di Erdos Renyi, che possiede due varianti. Nella prima si fornisce in input il numero di nodi e la probabilità che tra due nodi scelti a caso ci sia un collegamento, nella seconda versione invece si fornisce sia il numero di nodi che il numero di archi totali che dovranno apparire nel grafo.
  Siccome nel primo caso non si può stimare esattamente quanti collegamenti verranno creati nel grafo nel progetto è stata utilizzata la seconda tecnica.

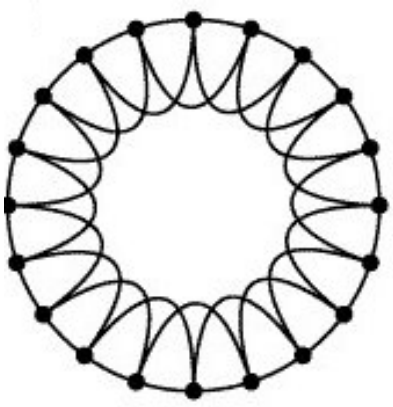


- scale-free, ovvero grafi che presentano un piccolo numero di hub, ovvero nodi che hanno molti più collegamenti rispetto agli altri vertici della

rete, che sono invece poco connessi.
In una rete scale-free il grado di distribuzione, ovvero la probabilità che un nodo selezionato casualmente abbia un certo numero di collegamenti, segue una legge di potenza (power law).

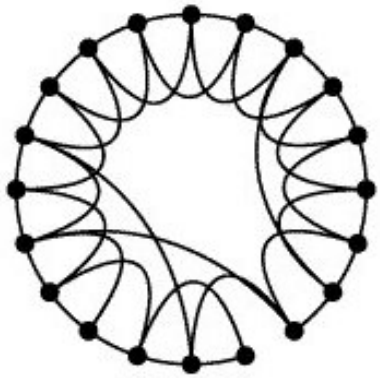


- regolari, ovvero grafi in cui ogni nodo ha lo stesso numero di vicini e quindi lo stesso grado. Un grafo regolare con nodi di grado k si chiama k-regular.



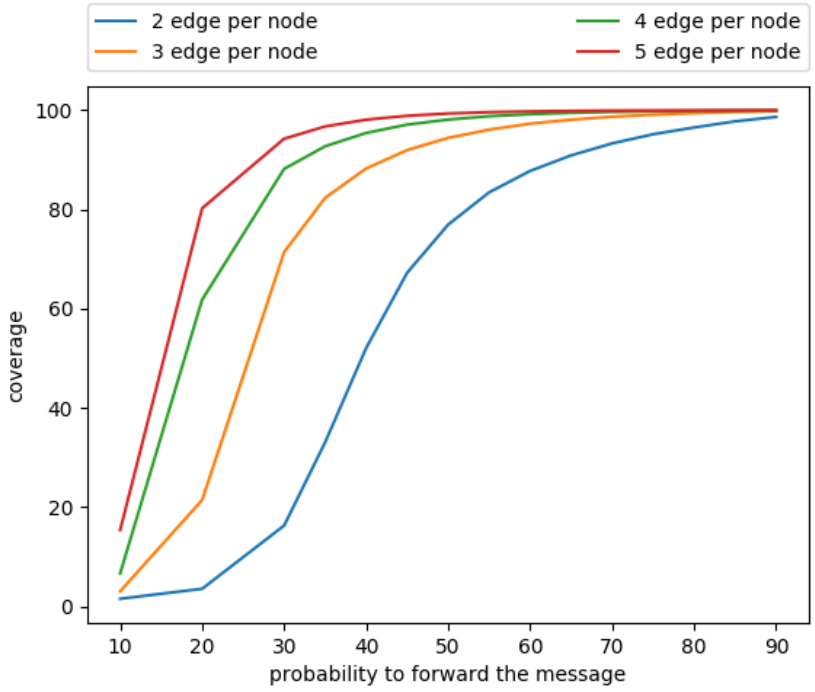
- small-world, ovvero grafi in cui la maggior parte dei nodi non sono vicini tra loro ma in cui molto probabilmente l'insieme dei vicini dei due nodi

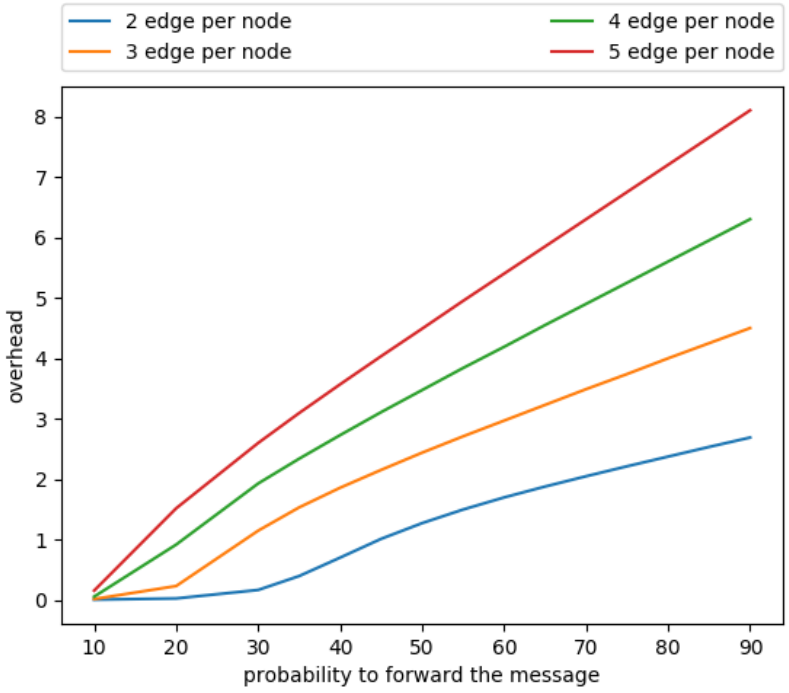
presenta intersezioni. Conseguentemente in questi grafi il numero di hop necessari per raggiungere due nodi scelti casualmente è parecchio basso. Questi grafi con la libreria igraph sono generati col modello di Watts-Strogatz.

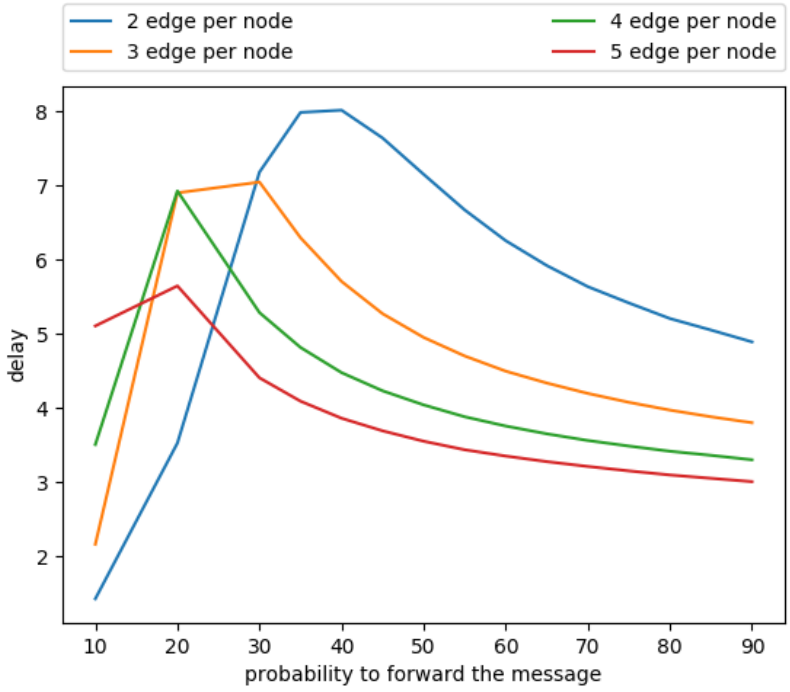


Si può constatare che fixed probability e probabilistic broadcast a parità di valore del parametro *prob* hanno risultati praticamente identici per quanto riguarda delay e overhead, mentre i valori di coverage per fixed probability sono sempre leggermente migliori.
Come prevedibile è emerso che aumentando il numero di archi nella rete il livello di coverage aumenta, il delay medio diminuisce ma l'overhead aumenta sensibilmente.

Nei seguenti grafici vengono riportati la coverage, il delay e l'overhead su grafi di 500 nodi generati casualmente, cambiando il numero di archi per nodo e usando l'algoritmo di gossip Fixed Probability:

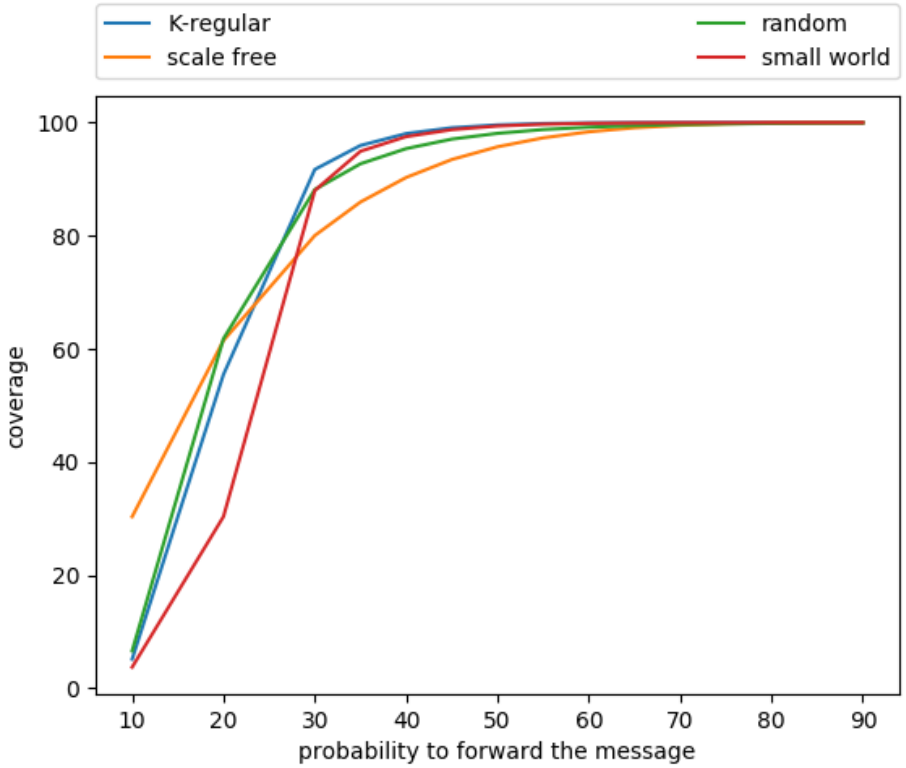

2 edge per node
3 edge per node
4 edge per node
5 edge per node
coverage
probability to forward the message

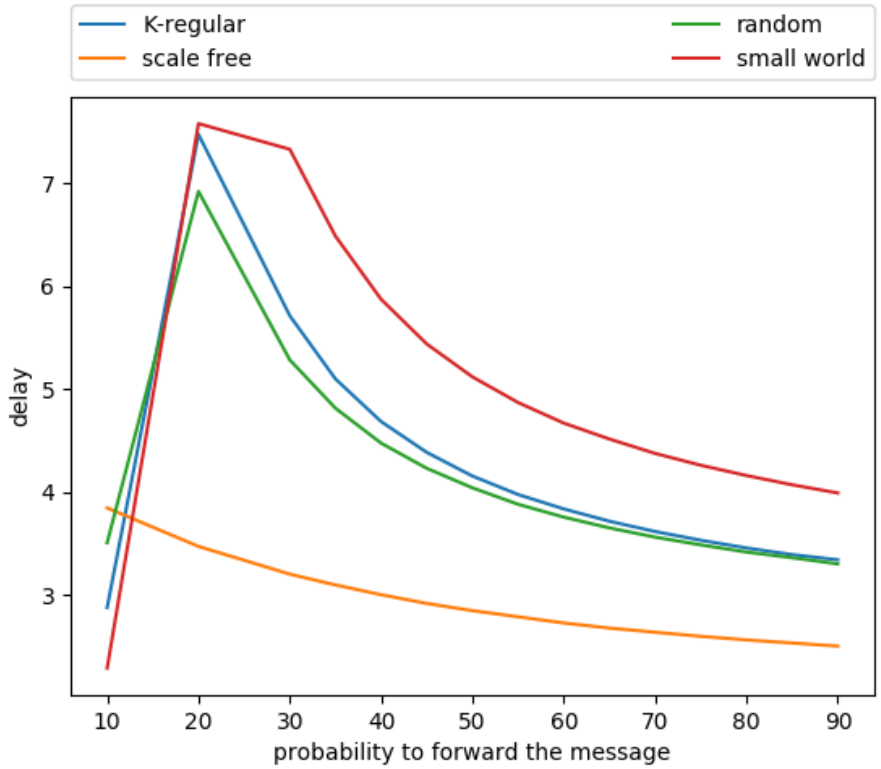

2 edge per node
3 edge per node
4 edge per node
5 edge per node
overhead
probability to forward the message

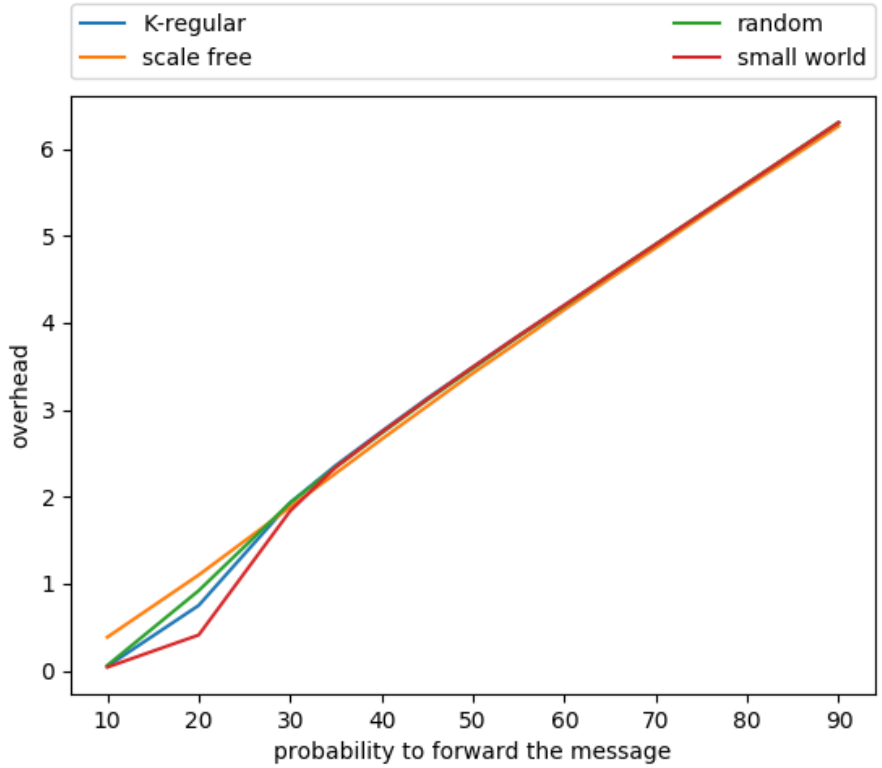


In una rete con più collegamenti il numero di messaggi inoltrati durante una simulazione è maggiore, favorendo il raggiungimento di un tasso più alto di coverage e riducendo il numero di hop medi per raggiungere un nodo. Allo stesso tempo però vengono mandati un numero più alto di messaggi non necessari. Si può arrivare dunque alla conclusione che, per un valore opportuno di *prob* (ad esempio per $prob \geq 35$), il numero di collegamenti per nodo nella rete è direttamente proporzionale a overhead e coverage raggiunti e inversamente proporzionale al delay medio.

Ora, verificato l'effetto del numero di archi sulle prestazioni, è stato analizzato l'impatto che ha la topologia di un grafo sulle performance della disseminazione. Di seguito sono riportati i grafici che mettono in confronto le prestazioni dell'algoritmo Fixed Probability eseguito sulle diverse topologie di rete. In ognuno dei 4 test i grafi sono costituiti da 500 nodi e 2000 archi.

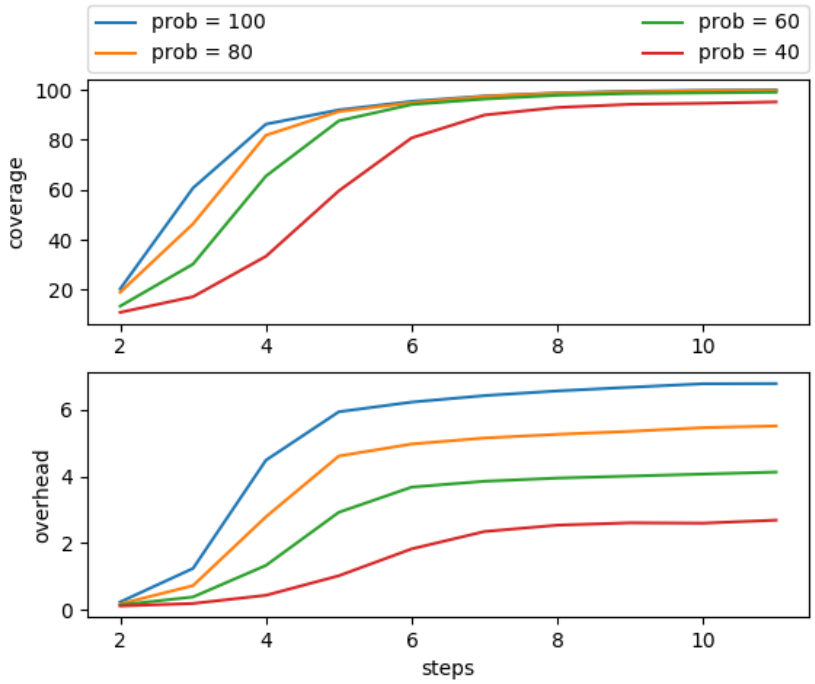

K-regular
scale free
random
small world
coverage
probability to forward the message


K-regular
scale free
random
small world
delay
probability to forward the message



Se quindi la topologia del grafo non ha grandi ripercussioni sul tasso di copertura raggiunto e sull'overhead non si può dire la stessa cosa sul delay: le reti scale-free presentano un livello di delay parecchio più basso, mantenendo sempre circa almeno un'unità di vantaggio rispetto ai grafi k-regular e randomici. Le reti small-world presentano invece un delay costantemente maggiore con un valore di $prob \geq 20$.
Dandelion, impostando il numero opportuno di step di fluff phase, riesce a raggiungere i livelli di coverage degli altri algoritmi ma, a parità di copertura, overhead e delay risultano estremamente più alti.
A parità di overhead infatti Dandelion risulta meno efficiente degli altri algoritmi. Ad esempio sulle reti scale-free con 4 collegamenti per nodo sia Dandelion con 5 passi di stem phase e 7 di fluff phase, sia Fixed Probability con $prob = 60$ hanno un overhead di circa 4.2, ma nel primo caso la copertura è del 76.8%, nel secondo invece è del 98.3%. Come ottimizzare Dandelion

quindi? L'idea è quella di non inviare i messaggi in broadcast durante la fluff phase ma di utilizzare Fixed Probability.

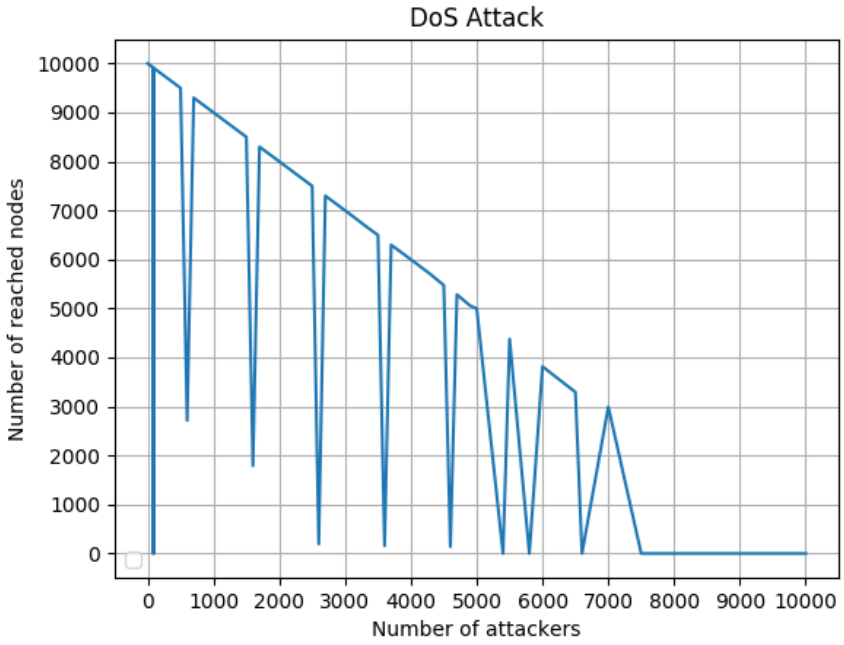


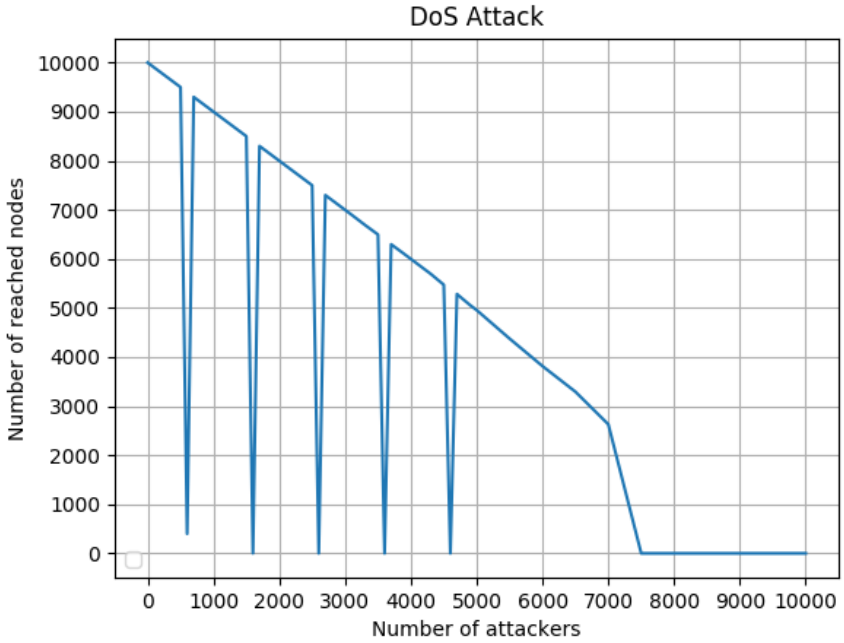
Come si evince dal grafico (ottenuto con test su grafi randomici con 500 nodi e 2000 archi), con almeno 6 step di fluff phase e assegnando un opportuno valore a *prob* per la fluff phase il livello di coverage è pressoché identico ma l'overhead cala sensibilmente.

Ad esempio ora la versione migliorata di Dandelion con *prob* impostato a 60 e 6 step di fluff phase ha un overhead di circa 3.7, molto simile all'overhead medio di Fixed Probability con $prob = 55$. Nel primio caso la copertura raggiunta è del 98.5%, nel secondo caso del 94.2%. Quindi, anche con l'ottimizzazione della fluff phase, Dandelion rimane sempre un algoritmo con performance peggiori rispetto a Fixed Probability, ma il calo delle prestazioni questa volta non è così drastico e può essere un compromesso accettabile per avere la garanzia dell'anonimità delle transazioni.

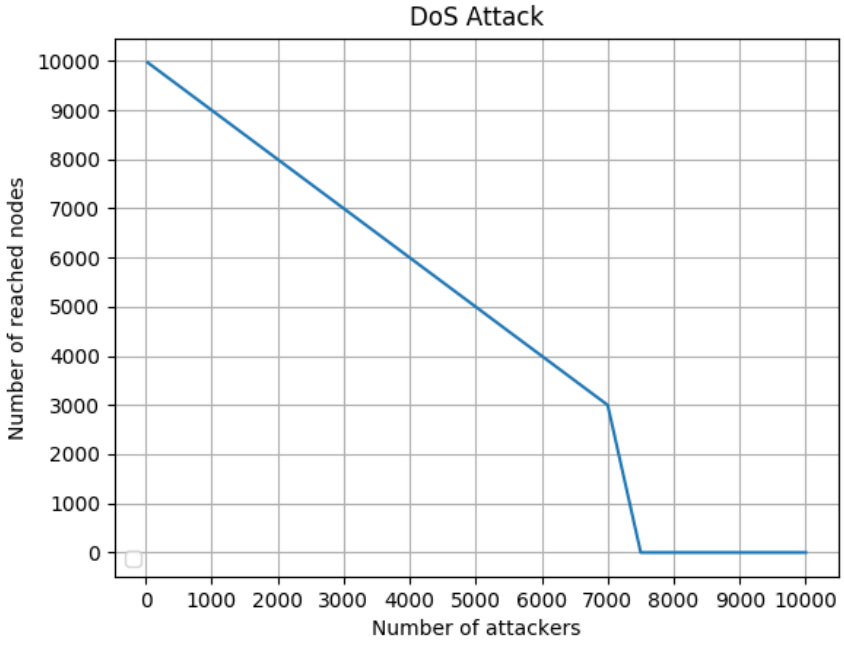
4.1 Analisi sui protocolli di gossip

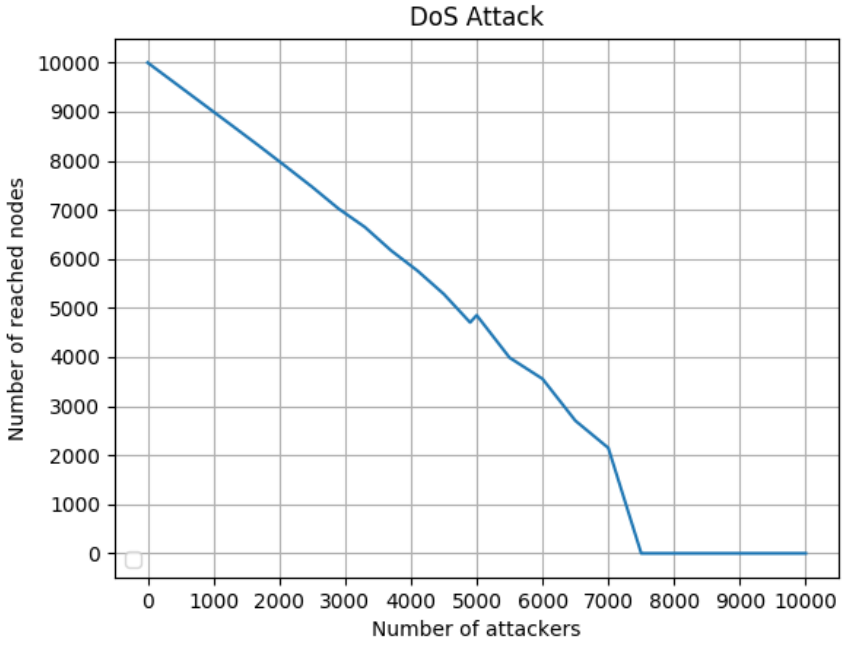
L'obiettivo di questi test è quello di verificare se e quanto la scelta dei vari algoritmi di gossip influisce sull'esito dell'attacco di denial of service contro la blockchain.
Nel simulatore le transazioni vengono diffuse nella rete tramite un algoritmo di gossip mentre i blocchi sono propagati in broadcast. È stato scelto quindi DoS nell'esperimento perché per altri attacchi sulla blockchain, come quello del 51% o il selfish mining, l'esito dell'attacco non è correlato al meccanismo di diffusione delle transazioni. Il TTL è stato impostato a 14 visto che 12 era il diametro massimo del grafo di 10000 nodi.
Dai test è emerso che la scelta del protocollo di gossip da utilizzare ha una certa influenza sugli esiti del DoS. Quello che influenza maggiormente i risultati non è però la scelta dell'algoritmo di disseminazione in sé, bensì la percentuale di coverage raggiunta con un determinato algoritmo. Usando un protocollo di gossip che garantisce una minore percentuale di coverage la possibilità di propagare un blocco da parte del nodo vittima cala, e può esserci il denial of service completo anche di fronte a una minore quantità di nodi malintenzionati. Ad esempio dai test effettuati su LUNES è venuto fuori che Dandelion con 3 step di stem phase(figura in alto) e Probabilistic Broadcast con $prob = 47$ (figura in basso) dovrebbero avere risultati simili per quanto riguarda la coverage.
I risultati di fronte a un attacco DoS sono conseguentemente simili.


DoS Attack
Number of reached nodes
Number of attackers
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

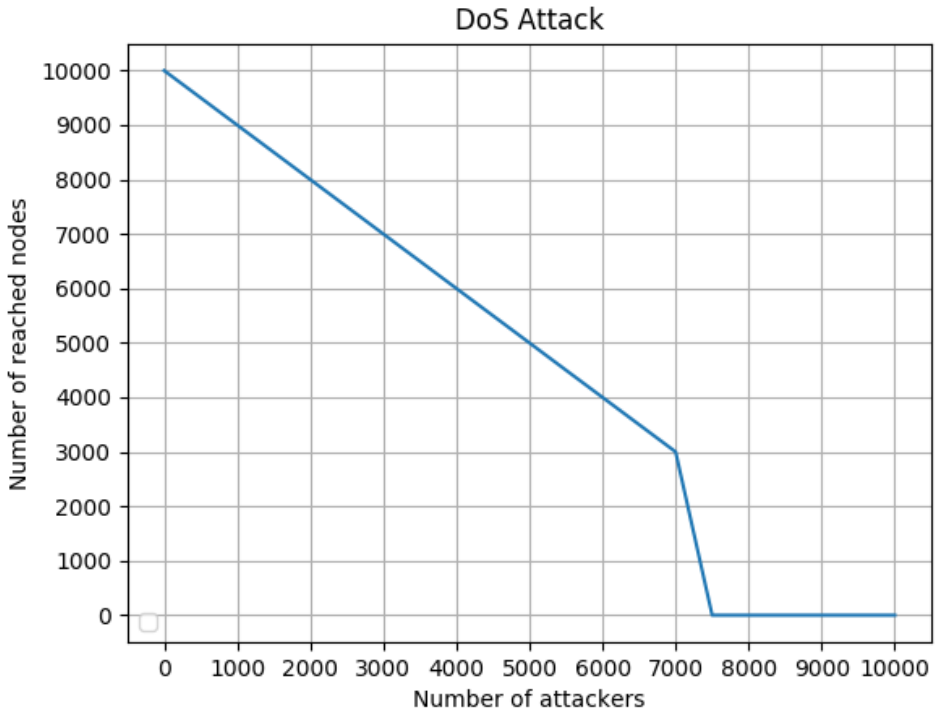

DoS Attack
Number of reached nodes
Number of attackers
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Si è deciso inoltre di ripetere i test su un diverso seed per la generazione di numeri pseudocasuali.



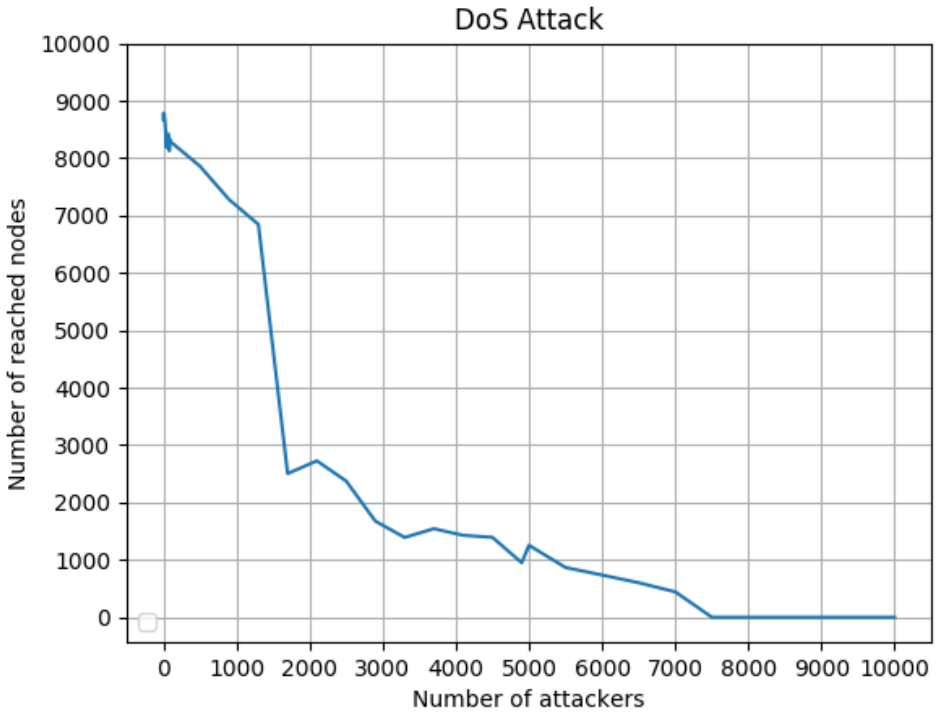


Bisogna notare che in questo caso, facendo eseguire più volte la stessa simulazione su seed diversi, è possibile ottenere risultati leggermente differenti. A volte in alcuni test si è osservato un comportamento seghettato, in cui si è raggiunto un DoS totale o quasi anche di fronte a un numero relativamente piccolo di nodi malevoli. Questo è sintomo del fatto che alcuni comportamenti probabilistici della simulazione possono influire sull'esito dell'attacco, come ad esempio la posizione dei nodi malintenzionati.
Di fronte a un algoritmo che garantisce una coverage molto alta (superiore al 98%) la scelta di come vengono propagate le transazioni non influenza l'esito dell'attacco: sia fixed probability con *prob* impostato a 80, sia Dandelion con 12 passi di fluff phase e 4 di stem phase (con il TTL impostato solo per questo test a 16 in modo da raggiungere una coverage molto alta) hanno riportato risultati identici (quelli della figura sottostante).



Utilizzando invece come esperimento un protocollo in grado di ottenere una

coverage molto bassa, come Fixed Probability con *prob*=25, i risultati sono i seguenti:



Gli utenti che ricevono le transazioni del nodo vittima sono molti meno in quanto, oltre alla presenza degli attaccanti, si aggiunge la scelta del protocollo di gossip a ostacolare la disseminazione della transazione. Da notare però che la soglia dopo la quale avviene un DoS completo rimane la stessa, ovvero intorno al 75% dei nodi della rete.

Capitolo 5

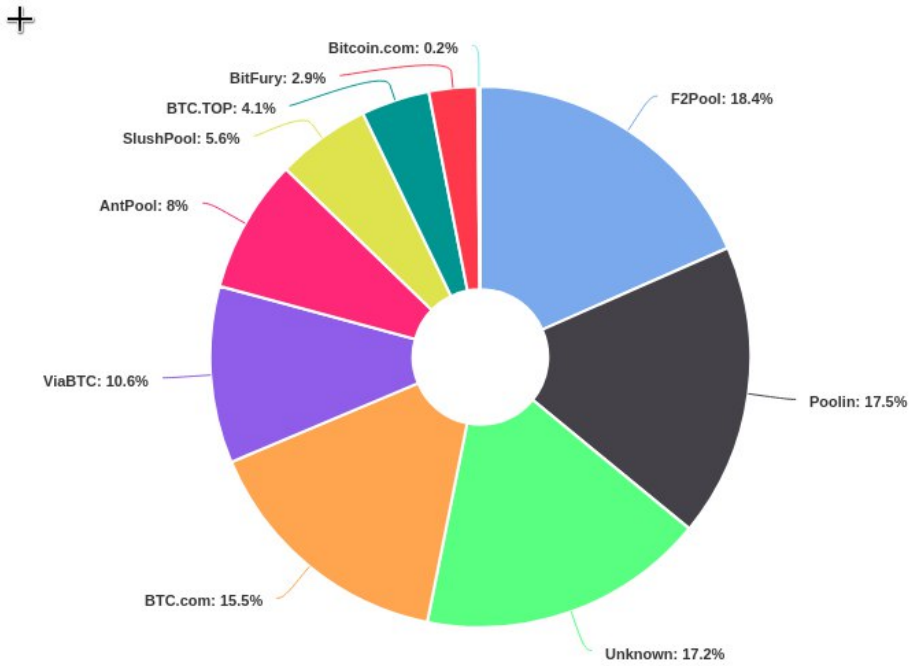
Quanto è realistico LUNES-bitcoin?

LUNES-bitcoin è un simulatore che mira a ricostruire le dinamiche computazionali e di diffusione dell'informazione che stanno alla base della blockchain di Bitcoin e simula la presenza di attori (i nodi della rete) che interagiscono all'interno dell'ambiente simulato. È inoltre possibile riprodurre alcuni tentativi di attacco informatico al sistema. Ovviamente molti aspetti del simulatore sono semplificati rispetto al comportamento originale, in quanto alcuni dettagli implementativi non sarebbero rilevanti nell'esito della simulazione. Ci sono però anche altri aspetti che, se implementati, possono contribuire a una maggiore veridicità della simulazione.

5.1 Hashrate dell'attaccante e mining pool

In LUNES-bitcoin ai miner presenti nella rete (escludendo per ora l'eventuale attaccante) viene assegnata casualmente una percentuale, che corrisponde alla frazione di potenza computazionale a disposizione del nodo (hashrate). Sommando gli hashrate di tutti i nodi si otterrà il 100%, ovvero il totale della potenza di calcolo crittografico della rete. Nel mondo reale però ai miner non conviene minare in solitaria ma è più efficiente per loro

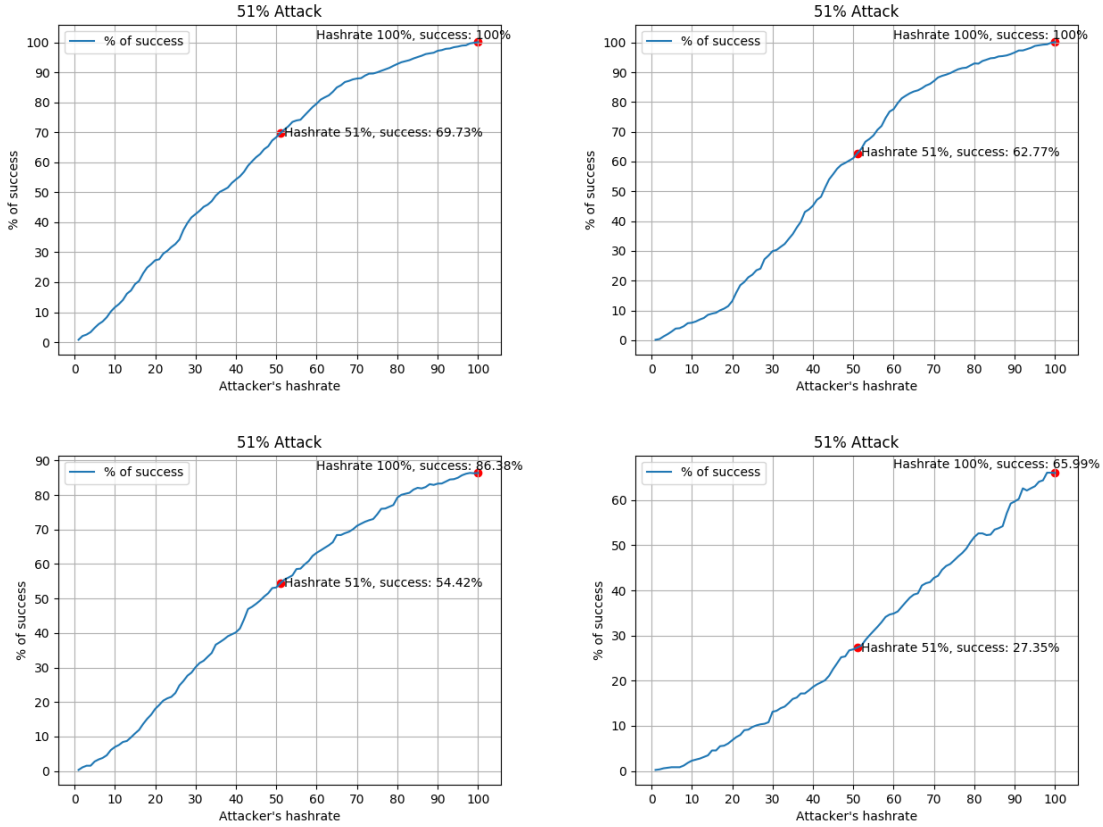
condividere la propria potenza computazionale con altri attori della rete, organizzandosi in pool e dividendosi oneri e ricompense, che verranno elargite in base alla quantità di potenza di calcolo crittografico con cui si è contribuito.
La distribuzione della capacità computazionale tra i vari pool della rete è all'incirca la seguente [2], anche se i dati non sono ufficiali ma sono solamente una ricostruzione:



Si è scelto di assegnare ai minatori singoli il 17,2% dell'hashrate totale della rete, corrispondente alla percentuale di miner non tracciata dalle statistiche, e probabilmente costituita almeno in parte da altri pool. I primi 9 nodi del grafo quindi rappresenteranno i pool sopraindicati.

Originariamente in LUNES-bitcoin la percentuale dell'hashrate dell'attaccante veniva sommata a quella degli altri nodi, con la conseguenza che l'hashrate totale della rete superava il 100%.

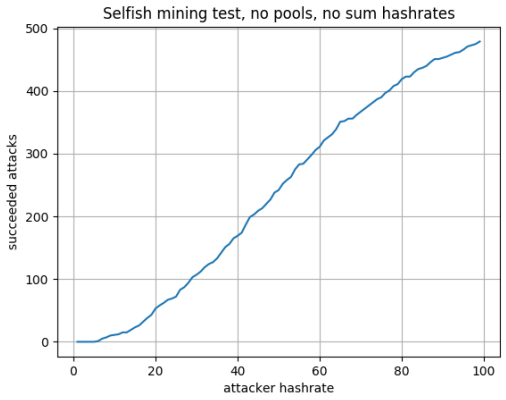
In base alle strategie per rappresentare i pool e la distribuzione dell'hashrate l'esito dell'attacco può essere differente. Si è deciso di testare quindi le varie configurazioni su una rete di 500 nodi con 4 collegamenti per vertice.
Nei grafici disposti in alto la percentuale dell'hashrate dell'attaccante non viene sommata a quella della rete, risultando quindi il totale come esattamente il 100%. Nelle figure in basso invece la somma di tutti gli hashrate risulterà (100 + x) %, in cui x è la potenza di calcolo del nodo malintenzionato rispetto al totale del sistema.
Inoltre nei grafici a sinistra c'è la presenza di 9 pool occupanti cumulativamente l' 82.8% della potenza computazionale, in quelli a destra tutti i miner agiscono singolarmente.

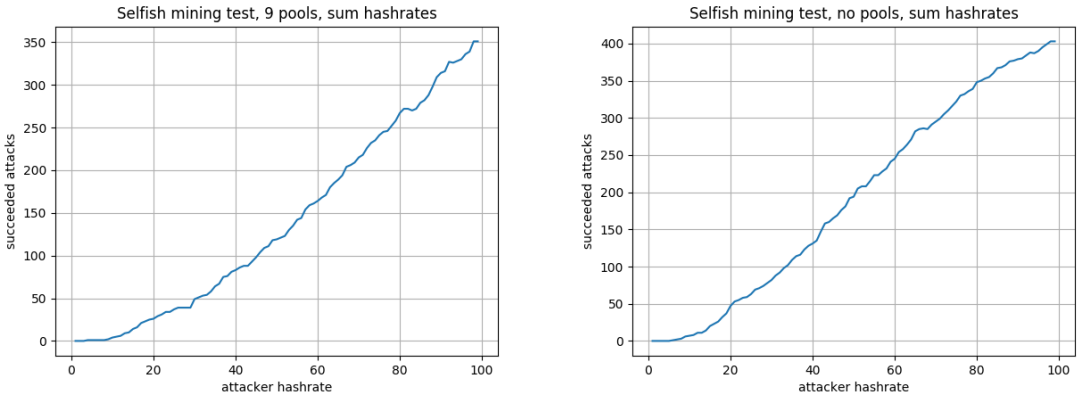


Come prevedibile nei grafici in basso la probabilità di successo è minore, in quanto l'attaccante può arrivare al più a pareggiare la capacità compu-

tazionale della rete, senza però riuscire a superarla e non potendo quindi raggiungere la fatidica soglia del 51%. Una simulazione in cui la percentuale dell'hashrate dell'attaccante non viene sommata al totale dovrebbe rivelarsi più realistica, sia perché va a testare effettivamente il caso in cui il nodo malintenzionato ha una capacità computazionale maggiore di quella di tutti gli altri nodi messi insieme, sia perché la difficoltà del puzzle virtuale da risolvere si adegua dinamicamente alle capacità di calcolo dei minatori, quindi un'improvvisa comparsa di un nuovo miner molto potente causerebbe il rapido aumento della difficoltà di trovare un hash corretto. Non avrebbe quindi senso che la somma delle percentuali degli hashrate superi il 100%.

Sono stati eseguiti anche gli stessi test per i selfish mining, in cui si considera che il nodo malintenzionato ha avuto successo nel caso la sua blockchain locale sia almeno due blocchi avanti rispetto a quella degli altri blocchi. Anche in questo caso gli esperimenti sono stati eseguiti su una rete di 500 nodi e 2000 archi. Come prima a sinistra si trovano i grafici in cui c'è la presenza di mining pool e in alto ci sono i grafici in cui l'hashrate dell'attaccante non viene sommato al resto della rete.





Anche in questo caso ovviamente nei grafici in alto l'attaccante ottiene risultati migliori, in quanto ha una potenza computazionale più altra rispetto ai rivali. C'è da notare che in questo caso la presenza di pool rende leggermente più difficile la riuscita dell'attacco.

5.2 Rapporto tra transazioni e blocchi

In LUNES-bitcoin un nodo, in ogni passo discreto della simulazione, tenta di minare un blocco e, successivamente, può inviare delle transazioni. Il numero delle transazioni inviate nel simulatore è simile a quello dei blocchi inviati, per cui il numero di transazioni incluse nei blocchi è molto basso, e può essere anche 0 in diversi casi. Normalmente in Bitcoin il numero di transazioni incluse in un blocco è compreso tra 800 e 2500 [3] ed è interesse dei miner includere un'alta quantità di transazioni in modo da riscuotere un numero maggiore di fee.

Bisogna però considerare il fatto che impostare un numero realistico di transazioni nella rete sarebbe particolarmente oneroso dal punto di vista della memoria, dato che a parità di step di simulazione la quantità di record da registrare sarebbe significativamente più alta. Inoltre questa modifica, per quanto semplice da implementare, non contruibuirebbe in modo significativo

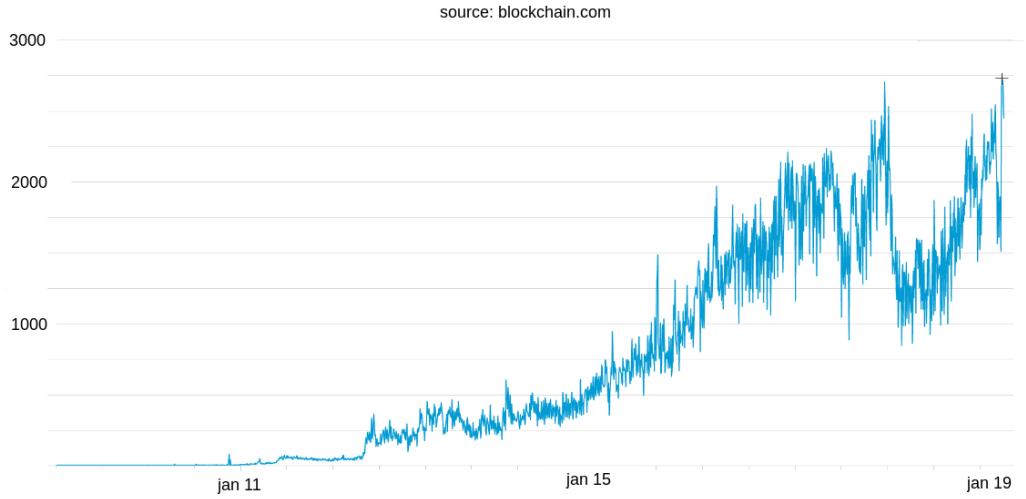


Figura 5.1: L'asse delle ordinate indica il numero medio di transazioni incluse nei blocchi

a rendere più realistici i test, in quanto per alcuni attacchi le transazioni sono ininfluenti (e vengono perciò disabilitate) e per altri invece non vi è una relazione stretta tra veridicità dei risultati e quantità di transazioni diffuse.

5.3 Durata di uno step della simulazione

In Bitcoin la difficoltà del puzzle criptografico da risolvere è modificata dinamicamente per far sì che, indipendentemente dalla potenza computazionale della rete, in media venga minato un blocco ogni 10 minuti.
In LUNES-bitcoin originariamente ogni passo discreto della simulazione rappresentava un minuto reale. Questo pone un problema: se da un lato vengono garantite tempistiche realistiche per quello che riguarda il mining, non si può dire altrettanto per ciò che riguarda l'inoltro dei messaggi. Ad esempio un messaggio che deve raggiungere un nodo a distanza 3 dal creatore impiega così 3 passi di simulazione per essere ricevuto. Realisticamente però ci vorrebbe molto meno di 3 minuti per ricevere un messaggio del genere. Le soluzioni per risolvere questo problema implicano l'aumento dei passi della

simulazione e la diminuzione conseguente della quantità di tempo media che viene rappresentata da uno step.
In particolare le idee per portare a termine questa modifica sono state due:

- aumentare il numero di step di n volte e conseguentemente incrementare di n volte la difficoltà di minare un blocco.
- aumentare il numero di step totali di n volte ed eseguire il tentativo di minare un blocco solo $1/n$ delle volte. In questo caso si potrà parlare di simulazione multilivello, in quanto differenti passi di simulazione rappresentano differenti quantità di tempo simulate. Ogni $n-1$ propagation step ci sarà un mining step.

Per ragioni sia di efficienza che di maggiore veridicità della simulazione è stato scelto di utilizzare la seconda strada. Si noti inoltre che se si prova a minare un blocco solo durante gli step in cui il modulo di un certo numero n è congruo a 0, allora è inutile scegliere un valore di n troppo grande, in quanto ci sarebbero diversi step della simulazione sicuramente vuoti in cui è impossibile che succeda qualcosa. Si è scelto dunque di impostare n come il valore del diametro massimo del grafo.

5.4 Catena unica

Una particolarità di Bitcoin e delle criptovalute che utilizzano la blockchain è il fatto che si possono verificare biforcazioni. Può infatti succedere che un nodo riceva più blocchi che hanno lo stesso blocco precedente e in caso si viene a creare un forking. Sarà la catena più lunga a sopravvivere, e i miner avranno interesse a proseguire tale catena in modo da ricevere il compenso economico pattuito. In LUNES-bitcoin tuttavia questo aspetto non veniva simulato ma ogni volta che un blocco veniva minato gli si assegnava un identificatore corrispondente alla posizione del blocco all'interno della catena.
Anche modificando il sistema in modo da ottenere le biforcazioni rimangono

comunque diversi aspetti decisamente semplificati rispetto alla strutturazione effettiva della blockchain di Bitcoin, dove le transazioni sono inserite in un Merkle tree e mantengono informazioni come il nonce o l'hash del Merkle root. Trattandosi di una simulazione però questi aspetti sono stati trascurati in quanto, a differenza del forking, non influiscono sull'esito dei test. Inoltre è preferibile memorizzare la minore quantità di informazione possibile, che è un aspetto importante in un progetto in cui l'impiego di memoria è molto consistente.

5.5 Fee

La fee è la quantità di denaro che il mittente di una transazione dona al minatore come premio per aver convalidato la transazione all'interno di una catena di blocchi valida. Il valore inserito come fee viene deciso dal mittente del pagamento e può essere anche 0. Più questo valore è alto, però, più il minatore ha interesse a inserire la transazione all'interno di un blocco in modo da ottenere una ricompensa maggiore. Una maniera di fare double spending potrebbe essere quella di effettuare due transazioni, una verso il nodo vittima in cui viene inserita una fee bassa o nulla e una verso se stessi (o verso un altro account controllato dall'utente malintenzionato), in cui si inserisce una fee molto più consistente. I minatori avranno dunque più interesse a inserire nel blocco che stanno provando a minare la seconda transazione, visto che guadagnerebbero di più dall'inserimento di quest'ultima. La prima invece verrebbe automaticamente trascurata anche in futuro, essendoci controlli che rendono impossibile spendere più volte lo stesso denaro (cioè il denaro generato dall'output della medesima transazione UTXO). Questi attacchi però difficilmente possono essere implementati su un simulatore di questo tipo, in quanto l'esito non dipende solo da motivazioni computazionali o facilmente simulabili.
È quindi ammissibile trascurare il fatto che inserire una fee più o meno cospicua può influenzare l'esito di alcune tipologie di attacchi.

Capitolo 6

Forking

Per ottenere una maggiore veridicità del sistema è stata cambiata la struttura dei blocchi, aggiungendo informazioni in modo da poter simulare effettivamente la presenza di più catene all'interno della blockchain.

```
typedef struct blockchain_element {
    int         id;          // ID of the block
    int         latesttrans; // ID of the lastest transacation (used as index)
    #ifdef FORKING
    int         prevId;      // ID of the previoud block in the chain
    int         position;    // position of the block in the blockchain
    #endif
    #ifdef TXDEBUG
    Transaction trans[500];  // Mean # of transactions per block (less is better for performance)
    #endif
} Block;
```

Oltre a *latesttrans*, che indica il numero di transazioni presenti all'interno del blocco e *id*, l'identificatore univoco del blocco, è stato aggiunto *position*, l'indicatore della posizione all'interno della catena. La posizione è definita come il numero di blocchi precedenti che si trovano percorrendo la catena a partire dall'origine più 1. Nella versione precedente sostanzialmente l'indice di un blocco nella sequenza di blocchi e il campo *id* avevano lo stesso significato, e corrispondevano con la posizione. L'altra aggiunta è stata l'inserimento del campo *prevId*, che fa riferimento all'identificatore del blocco precedente della blockchain. L'ID del blocco precedente rende possibile, a partire da qualsiasi nodo, risalire fino alla radice. Il campo *position* è stato inserito con

il fine di sapere la lunghezza di una catena in un determinato punto, informazione che risulta utile sia per verificare la correttezza del funzionamento del sistema, sia per risparmiare computazioni quando è necessario ottenere l'informazione.
Sono state effettuate inoltre delle piccole aggiunte alla parte statica delle simulated entity, aggiungendo un array di blocchi, *heads*, ovvero i nodi che stanno in cima a una catena. L'idea è di avere un array di dimensione limitata: non sarà infatti un problema perdere traccia delle catene più corte e ormai diventate obsolete.

```
/*! \brief Static part of the SE state */
typedef struct static_data_t {
    #ifdef FORKING
    Block     heads[HEADS_LENGTH];                    // longest head of the block
    #endif
    char      changed;                        // ON if there has been a state change in the last timestep
    char      freerider;                      // 1 if free-rider, 0 not free-rider
    float     time_of_next_trans;             // Timestep in which the next new transaction will be created and sent
    float     time_of_next_check;             // Timestep in which the next check for new block will be created and sent
    Block     blockchain[2500];               // Array of blocks
} static_data_t;
```

Inoltre con l'aggiunta del forking può essere necessario aumentare il numero di blocchi memorizzabili nella blockchain. Infatti in una simulazione composta da 5000 passi discreti si riesce a ottenere una catena principale con una lunghezza di circa 1450 unità. Però il numero totale di blocchi minati, e quindi conservati in locale, è quasi 2300, ben oltre a 1500 che era la precedente dimensione del buffer.

La logica utilizzata per gestire le biforcazioni è la seguente:

- Quando un nodo mina un nuovo blocco lo inserisce nella catena più lunga, aggiungendolo alla blockchain e andando a sostituire il blocco precedente tra i blocchi-testa. L'identificatore che viene assegnato è un numero casuale generato tra 0 e MAXINT (2147483647). Le collisioni sono dunque molto improbabili.

- Quando un nodo riceve un blocco B, se questo non è presente nella blockchain allora viene inserito. Inoltre:

- Se B non ha un blocco precedente e dunque è in posizione 1 nella catena, allora si inserisce B tra i blocchi-testa.

- Se P, il blocco precedente di B, è tra i blocchi-testa (e quindi presente nella blockchain) allora si sostituisce P con B tra i blocchi-testa.

- Se il blocco precedente P è nella blockchain ma non nei blocchi-testa allora si va a vedere se per caso esiste già un blocco N che sia il successivo di B. Se non esiste nessun blocco successivo di B nella blockchain, allora si inserisce B tra i blocchi-testa, altrimenti si inserisce nei blocchi-testa l'ultimo successore di B. Per capire se esiste un blocco che sia il successore di B si itera nella blockchain alla ricerca di un blocco che nel campo *idprev* abbia l'ID di B.

- Se il blocco precedente P non è presente nella blockchain allora si fa una richiesta tramite messaggio di Askblock per tutti i nodi la cui posizione è maggiore o uguale a quella di B - 2 (sapendo di aver bisogno di almeno del precedente del blocco precedente). In alternativa, assumendo che n sia la posizione del nodo-testa della catena più lunga e m quella di B chiederò tutti i blocchi in posizione maggiore o uguale a n-1 nel caso $n < m - 2$, in quanto sicuramente mancherebbero nella blockchain dei nodi con posizione minore di m-2. B non verrà aggiunto ai blocchi-testa fino a che non sarà collegato a una catena ma viene comunque inserito nella blockchain in modo da non dover richiedere B in futuro.

Per analizzare il comportamento della blockchain simulata ci si avvale di file di log, nei quali viene stampato ogni messaggio ricevuto o inviato relativo a blocchi o transazioni (queste possono essere disabilitate per alleggerire il programma laddove non influenzano l'esito della simulazione). Le informazioni salvate nei log possono essere dei seguenti tipi:

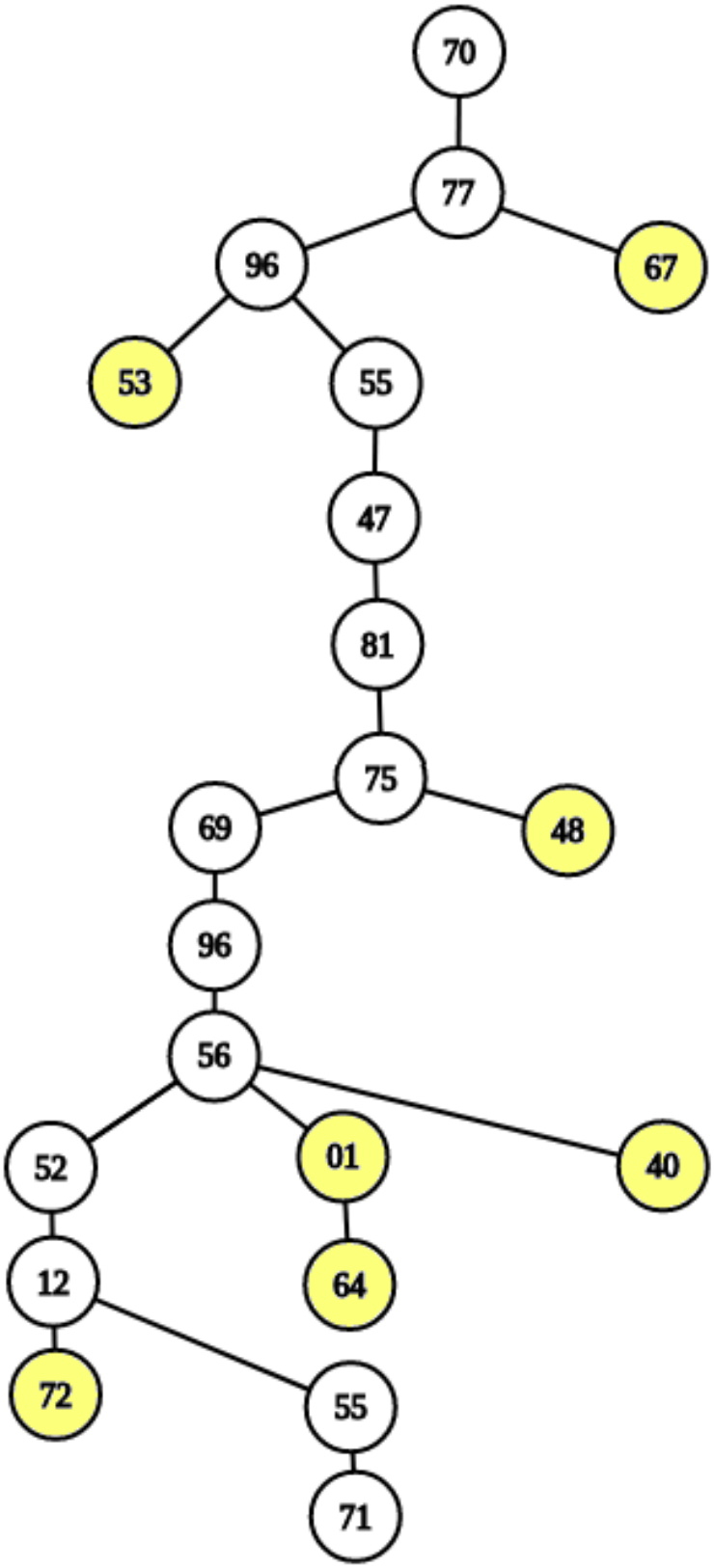
- BS (Block Sent): viene creato un blocco e inoltrato a tutti i vicini.

- BR (Block Received): un nuovo blocco viene ricevuto e inviato ai vicini.
- TS (Transaction Sent): viene creata una transazione e inviata a tutti i vicini.
- TR (Transaction Received): una transazione viene ricevuta e conseguentemente inoltrata agli altri nodi.
- BRS (Block Received but Stale): viene ricevuto un blocco stale.
- TRS (Transaction Received but Stale): viene ricevuta una transazione già conosciuta.
- ABS (Ask Block Sent): viene chiesto ai vicini il contenuto di un blocco, indicando la posizione della catena a cui si è arrivati.
- ABR (Ask Block Received): un messaggio di AskBlock viene ricevuto, seguirà dunque l'invio dei blocchi al nodo richiedente.

Dai test effettuati (senza attacchi) non si è però mai verificato il caso in cui un nodo avesse bisogno nel corso dell'esecuzione di inviare un messaggio di AskBlock, quindi per testare questa casistica si è deciso di escludere un nodo dalle ricezione dei blocchi per i primi 30 step della simulazione. Qui si è notato un picolo problema: nonostante il corretto funzionamento del sistema (determinato dal fatto che il nodo era stato in grado di recuperare tutti i blocchi non ricevuti nel corso di soli due step nella simulazione) il numero di messaggi AskBlock inviati e ricevuti era considerevole. Fino allo step 30 erano stati minati 9 blocchi, con la catena più lunga che aveva raggiunto la lunghezza di 7 unità. Per recuperare quei blocchi sono stati necessari 10 messaggi di AskBlock, che a loro volta hanno generato 81 messaggi di risposta (ABR). Basandosi sulle tecniche impiegate dai protocolli di gossip per le transazioni si è deciso quindi, attraverso una fonte randomica, di inviare i messaggi di AskBlock solo a una frazione dei nodi vicini. Impostando questa percetuale al 50% il numero di AskBlock è stato ridotto a 3 e le risposte a 23.

Si noti che se per coincidenza non si dovesse mandare il messaggio di AskBlock a nessuno dei vicini, comunque la ricezione di un nuovo blocco innescherebbe di nuovo il meccanismo dell'AskBlock, per cui non c'è pericolo che col passare del tempo un nodo non riceva tutti i blocchi della catena principale. Nonostante questo miglioramento possa sembrare trascurabile, in altri casi, come in presenza dell'attacco di selfish mining, permette di risparmiare una notevole quantità di messaggi.

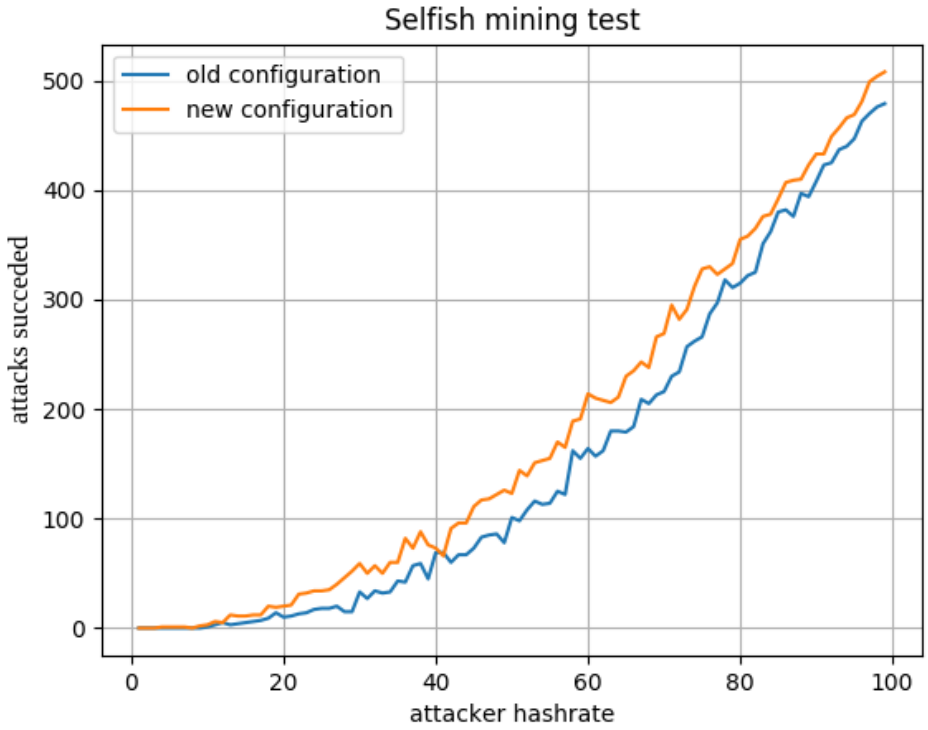
Al termine dell'esecuzione il numero dei blocchi minati che viene incluso all'interno della catena principale è di circa 2/3. Ad esempio eseguendo un test con 100 nodi (di cui il 60% di minatori) dopo 5000 passi di simulazione si hanno 2189 blocchi minati e la catena principale lunga 1421 unità.
Sotto si riporta la configurazione della blockchain dopo 60 passi di esecuzione, i numeri all'interno dei nodi nel grafico rappresentano le ultime due cifre dell'ID del blocco. I blocchi minati durante quegli step sono 21 e si crea una catena principale con 14 elementi. I blocchi rappresentati in giallo sono quelli che rimarranno esclusi dalla catena principale e perciò tutte le transazioni presenti in essi non saranno considerate valide.


70
77
96
67
53
55
47
81
75
69
48
96
56
52
01
40
12
64
72
55
71

6.1 Selfish mining

È considerato andato a buon fine un attacco di selfish mining quando l'attaccante, nella sua catena personale, riesce a essere in vantaggio di due blocchi rispetto alla catena principale della blockchain pubblica. Una volta raggiunto lo scopo l'attaccante diffonde i blocchi attraverso la rete e ricomincia a minare per cercare di portarsi in vantaggio.
In uno scenario realistico invece un attaccante potrebbe continuare ad accumulare blocchi senza pubblicarli raggiungendo un vantaggio anche superiore alle due unità. Portarsi troppo avanti nella catena locale però potrebbe palesare le intenzioni del nodo malintenzionato all'intero sistema e quindi potrebbe non essere un comportamento produttivo.

Nella figura seguente vengono mostrati il numero di attacchi portati a termine da parte dell'attaccante. I blocchi presenti nella blockchain che sono stati minati dal nodo malintenzionato saranno sempre maggiori o uguali in numero al doppio dei tentativi di attacco che hanno avuto successo (in quanto a ognuno di questi corrisponde la pubblicazione sulla rete di due blocchi).
Per l'ambiente di test si è deciso di utilizzare un grafo di 500 nodi e 2000 collegamenti, facendo girare la simulazione per 5000 step. Siccome una delle possibili critiche a Lunes-bitcoin era le gestione dei timestep si è deciso di testare sia la versione (old version) in cui in ogni passo di simulazione un nodo può minare un blocco, sia il nuovo approccio multilivello (new version) in cui solo ogni n step un nodo può provare a minare un blocco e in cui gli step totali sono n volte quelli originali. Il valore scelto di n è stato 8 in quanto appena superiore al diametro massimo del grafo.

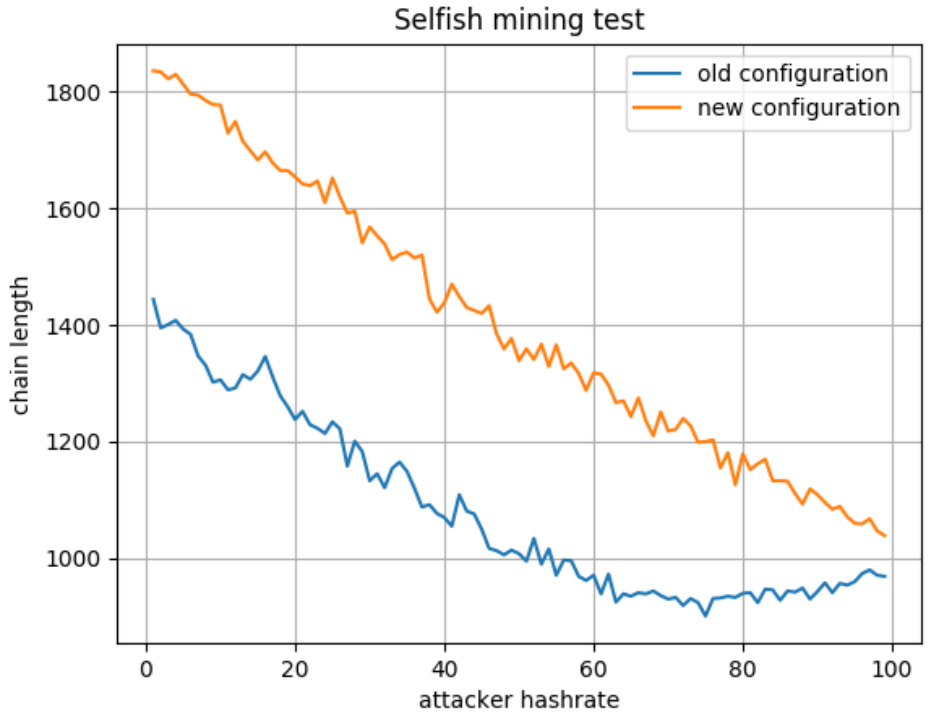


I tentativi di selfish mining fanno sì che, in uno stesso intervallo di tempo, vengano minati meno blocchi rispetto a una situazione in cui c'è assenza di comportamenti malevoli. Infatti quando l'attaccante ha successo il lavoro degli altri nodi viene momentaneamente vanificato, mentre quando l'attaccante fallisce si crea la situazione in cui vengono minati dei blocchi che però non vengono mai comunicati nella rete.

Per quanto riguarda l'impatto del nuovo meccanismo di gestione dei timestep si può notare che con la nuova versione il numero di tentativi di selfish mining portati a termine con successo è leggermente ma costantemente maggiore. Questo succede probabilmente perché con la nuova configurazione vengono aggiunti più blocchi nella catena principale. Il motivo della maggiore facilità di allungare la catena principale è dovuto alla minore presenza di conflitti tra blocchi nella stessa posizione. Infatti prima era possibile che un nodo creasse

e distribuisse un blocco senza aver prima ricevuto l'ultimo blocco minato nel sistema, andando conseguentemente a creare biforcazioni.
Si supponga ad esempio che due nodi della rete X e Y si trovino a distanza n, e che X crei un blocco al passo t della simulazione. Se Y riuscirà a minare un blocco in uno step temporale compreso tra t e $t + n$, allora lo diffonderà attraverso la rete senza aver prima ricevuto il blocco creato da X, creando così una biforcazione. Con il nuovo meccanismo di gestione dei timestep questo può avvenire solamente se entrambi i nodi minano il blocco nello stesso passo della simulazione.

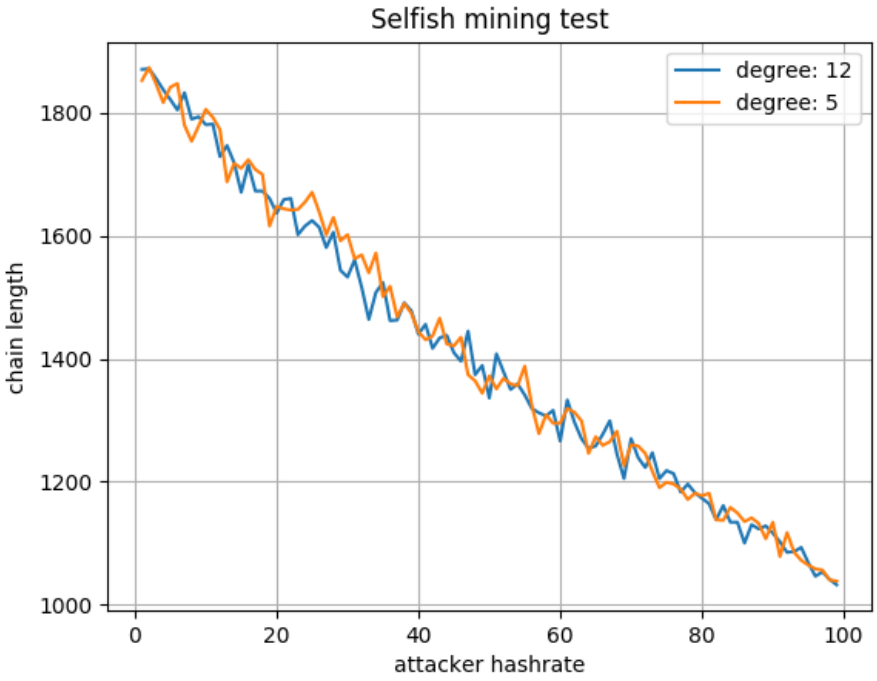
Il seguente grafico mostra invece la correlazione tra l'hashrate del nodo malintenzionato e il numero di blocchi che costituiscono la catena principale.

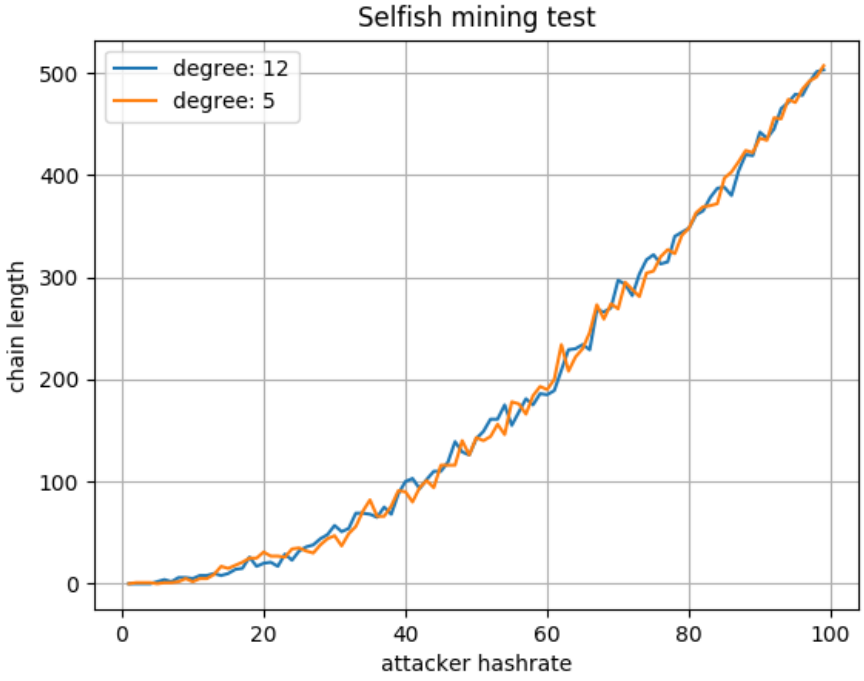


Si può notare che a causa dei vari eventi non deterministici il comportamento del grafico non è lineare ma la tendenza che associa l'aumento dell'hashrate

dell'aggressore con la diminuzione della lunghezza della catena principale è piuttosto marcata. Il comportamento scorretto dunque danneggia la produttività di blocchi complessiva del sistema.
In questo grafico l'impatto del nuovo meccanismo di gestione degli step discreti di simulazione è evidente. Anche in questo caso il fatto che che i blocchi abbiano il tempo di essere propagati prima che si possano verificare altre operazioni di mining rende più probabile che un blocco venga inserito nella catena principale. Le biforcazioni però, seppur in maniera minore, avvengono lo stesso, sia perchè è comunque possibile che due blocchi vengano minati nello stesso passo della simulazione, sia per l'effetto del nodo attaccante.

Si è inoltre voluto testare l'impatto della posizione del nodo attaccante all'interno della rete. Nel grafo il numero di vicini a cui è collegato un nodo è variabile, e può assumere un valore compreso tra 5 e 12 (con una media di 8). Si è deciso dunque di testare i due estremi, ovvero impostando come attaccante sia un nodo con il numero minimo di collegamenti, sia uno con il numero massimo di archi uscenti.


Selfish mining test
degree: 12
degree: 5
chain length
attacker hashrate
1000
1200
1400
1600
1800
0
20
40
60
80
100


Selfish mining test
degree: 12
degree: 5
chain length
attacker hashrate
0
100
200
300
400
500
0
20
40
60
80
100

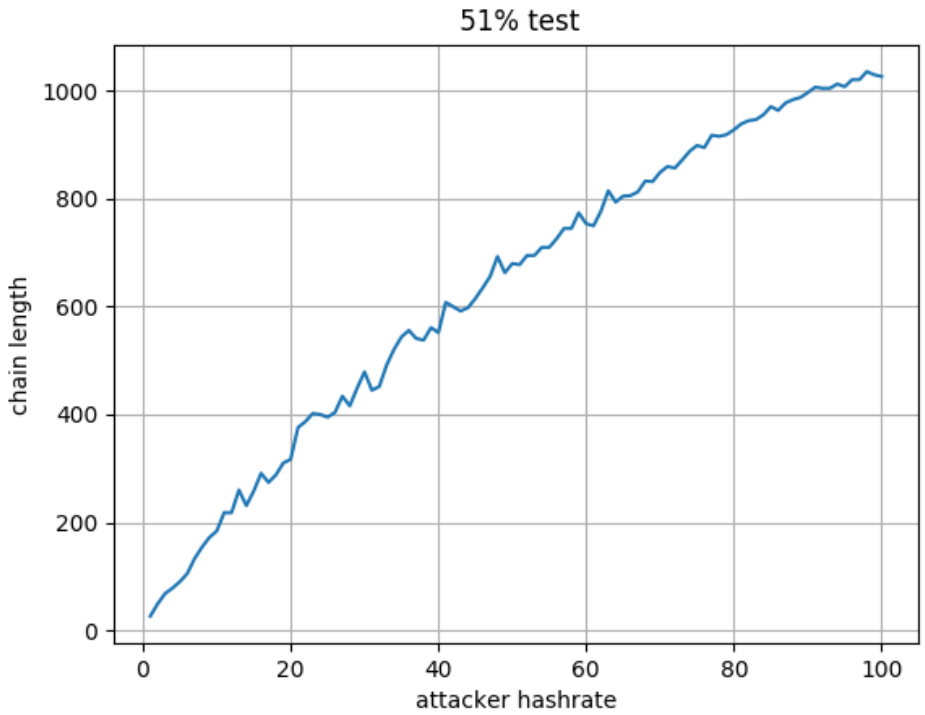
Si può osservare che il numero di collegamenti del nodo attaccante nel grafo non è un fattore infuente, e sia il numero totale di attacchi riusciti che la lunghezza della catena principale non dipendono strettamente da questa impostazione.

6.2 51%

Per testare la capacità dell'attaccante di controllare la genesi dei blocchi della blockchain si è deciso di ricostruire la catena principale in modo da capire:

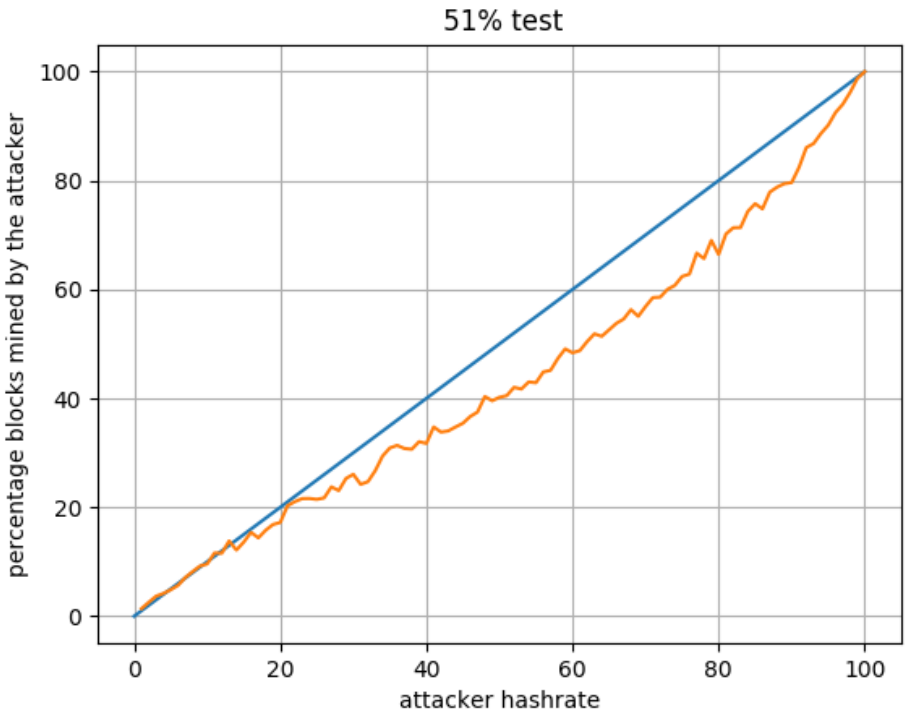
- quanti blocchi sono stati effettivamente minati dal nodo malintenzionato e inseriti nella catena principale.
- in percentuale quanti tra i blocchi della catena principale sono stati minati dall'attaccante
- in percentuale quanti tra i blocchi minati dall'attaccante finiscono nella catena principale.

I test sono stati eseguiti su un grafo di 500 nodi e 2000 archi ed è stata utilizzata la nuova versione della gestione dei timestep, cioè quella in cui ci sono n * 5000 passi e solo ogni n passi è possibile che venga minato un blocco.

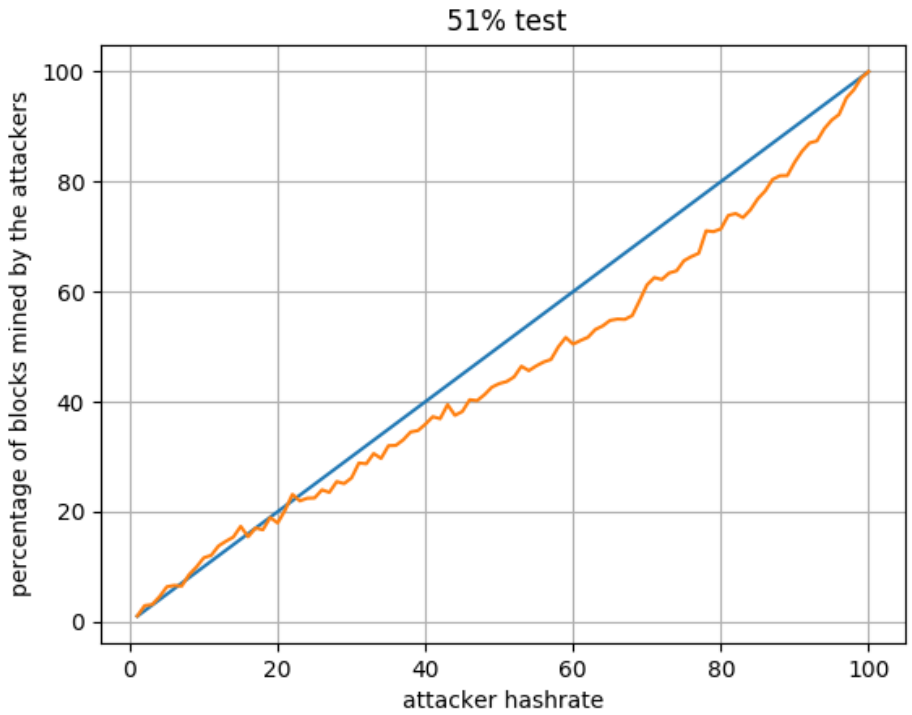


L'asse delle ascisse indica la percentuale dell'hashrate dell'attaccante, l'asse delle ordinate indica il numero di blocchi nella catena principale della blockchain che sono stati minati dall'aggressore.

Come prevedibile al crescere dell'hashrate a disposizione del nodo malitenzionato cresce anche il numero di blocchi inseriti nella catena principale che sono stati minati dall'attaccante. Questa è la logica conseguenza del fatto che più potenza computazionale si ha a disposizione più è facile minare un blocco.

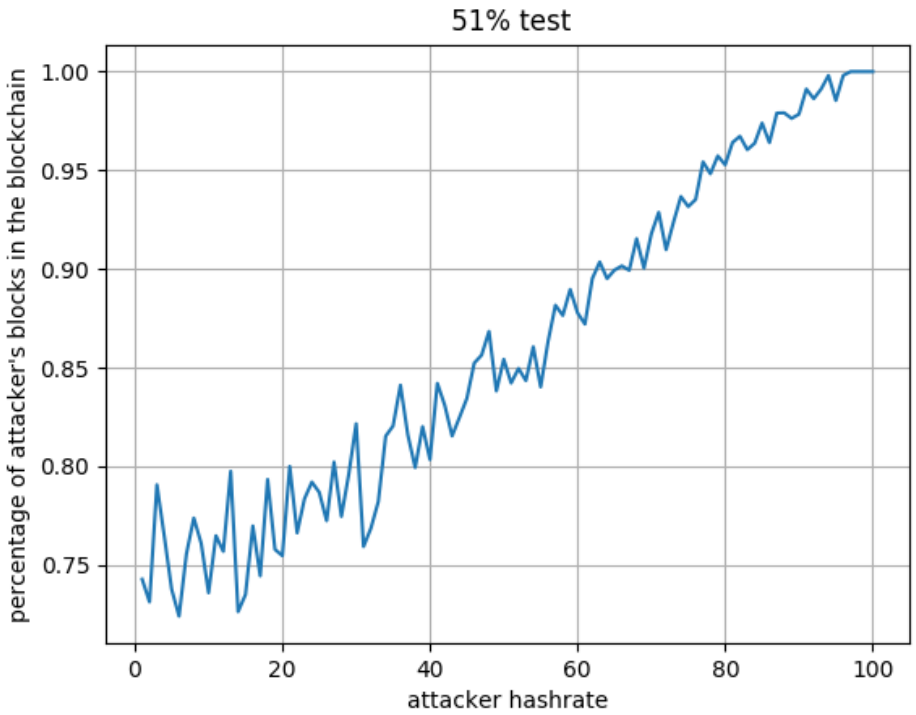


Osservando il grafico appena sopra si può notare che, come era prevedibile, nella catena principale la percentuale di blocchi che sono stati minati dall'attaccante (curva arancione) è direttamente proporzionale all'hashrate di quest'ultimo. Tuttavia la relazione non è perfettamente lineare ma, costantemente dal 20% di hashrate in poi, l'attaccante inserisce nella catena principale una percentuale di blocchi che è minore della percentuale della potenza di calcolo della rete a disposizione. Questa tendenza è particolarmente accentuata quando l'hashrate è compreso tra il 50% e il 90%, dove la differenza tra il comportamento atteso (indicato con la linea blu) e quello effettivo supera il 10%.
Per assicurarsi che il risultato ottenuto non fosse frutto dalle casualità si è scelto di creare un nuovo seed per la generazione di numeri casuali e di testare l'identica configurazione con il nuovo seed.



Anche in questo caso l'andamento del grafico è però molto simile, con picchi vicini al 10 % di differenza tra il comportamento atteso e quello ottenuto dal test. Anche in questo caso, per un hashrate dell'attaccante compreso tra il 30% e il 95%, la percentuale di blocchi minati dal nodo malintenzionato è sempre inferiore al suo hashrate, ovvero al comportamento atteso. Tale caratteristica dunque non si può attribuire alla casualità.

Nel grafico sottostante si può osservare invece quale sia la percentuale dei nodi minati dall'attaccante che finisce effettivamente nella catena principale. Più questa percentuale è alta più il nodo malintenzionato ha possibilità di controllare i dati inseriti nella blockchain ed effettuare double spending.



Anche in questo grafico all'aumentare del valore delle x tendenzialmente aumenta anche il valore delle y, ma l'impatto delle scelte probabilistiche del sistema fa sì che non ci sia un comportamento strettamente lineare.

Conclusioni

Un risultato interessante sulla prima parte degli esperimenti è stato che Dandelion, indipendentemente dalla topologia del grafo rappresentante i nodi della rete, garantisce performance peggiori rispetto a quelle dei protocolli di disseminazione standard. Il divario tra Dandelion e Fixed Probability o Probabilistic Broadcast però può essere ridotto combinando i due algoritmi, ovvero usando uno dei due protocolli anziché il broadcast puro durante la fluff phase. La differenza fra le prestazioni di Dandelion e quelle degli altri protocolli può essere quindi un prezzo opportuno da pagare per ottenere una maggiore garanzia sull'anonimità del mittente delle transazioni.
Per quanto riguarda l'impatto della topologia del grafo sulle prestazioni dei protocolli si può stabilire che all'aumento delle connessioni tra i nodi crescono coverage e overhead, mentre diminuisce il delay medio. Le reti scale free inoltre a partità di coverage e overhead consentono di avere un delay significativamente inferiore rispetto alle altre conformazioni del grafo, e in particolare rispetto alle reti small world.
Inoltre i risultati dei test sui protocolli Fixed Probability e Probabilistic Broadcast possono variare notevolmente in base al parametro *prob* utilizzato, che indica la probabilità di inoltrare un messaggio a un vicino o a tutti i vicini. Diminuendo il valore di *prob* si riesce a ridurre delay e overhead ma si rischia anche di peggiorare la copertura complessiva dei messaggi.
Dai test è emerso però che con un valore di *prob* di circa il 60% si garantisce una copertura pressoché completa, risparmiando dunque l'invio di messaggi inutili rispetto alle versioni con un valore maggiore di *prob*.

Con la nuova configurazione di LUNES-bitcoin (forking e approccio multilivello) è stato possibile ricostruire più realisticamente il comportamento della blockchain.
Anche la maniera di valutare l'esito di alcuni attacchi differisce col nuovo approccio. Se prima era stato necessario definire l'attacco del 51% come portato a compimento nel caso in cui i blocchi dell'attaccante fossero stati ricevuti da almeno la metà dei nodi, ora si è potuta verificare effettivamente la capacità di un nodo dato un certo hashrate di inserire i propri blocchi all'interno della catena principale della blockchain.
L'approccio multilivello invece è stato necessario a evitare un numero eccessivo di biforcazioni, che si potevano verificare a causa dei ritardi di propagazione dei blocchi. Infatti il tempo per diffondere un messaggio è abbondantemente inferiore a 1 minuto, che è la quantità di tempo simulato durante uno step di simulazione.
Un altro cambiamento ha riguardato gli hashrate dei nodi, che vengono ridimensionati proporzionatamente alla potenza di calcolo dell'attaccante, in modo che la somma degli hashrate di tutti i nodi, attaccante compreso, sia sempre del 100%. Nella versione precedente l'hashrate del nodo malintenzionato veniva sommato a quello della rete, per cui non era possibile simulare la situazione in cui la potenza di calcolo dell'attaccante era superiore a quella di tutti gli altri nodi messi insieme.
I test sui possibili attacchi su Bitcoin hanno dimostrato, come prevedibile, che all'aumentare della potenza di calcolo a disposizione dell'attaccante aumentano conseguentemente anche le possibilità dell'utente malintenzionato di controllare quali blocchi e transazioni vengono inseriti nella catena principale della blockchain.
È stato verificato invece che non è rilevante la posizione dell'attaccante all'interno della rete. Infatti l'esito degli attacchi è stato pressoché identico indipendentemente dal numero di connessioni del nodo malintenzionato.

Un risultato interessante inoltre è stato constatare che, in presenza di un miner con a disposizione più del 20% della potenza computazionale del sistema, la percentuale di blocchi inseriti nella catena principale della blockchain da tale nodo minatore sarà inferiore al suo hashrate.

Bibliografia

[1] Neudecker, Till, Philipp Andelfinger, and Hannes Hartenstein. "Timing analysis for inferring the topology of the bitcoin peer-to-peer network"

[2] https://www.blockchain.com/en/pools

[3] https://cryptoslate.com/bitcoin-transactions-per-block-at-all-time-highs/

[4] https://news.bitcoin.com/the-number-of-cryptocurrency-wallets-is-growing-exponentially/

[5] https://bitnodes.earn.com/

[6] https://digiconomist.net/bitcoin-energy-consumption

[7] https://news.8btc.com/no-more-than-60-miners-remain-in-china-80-of-them-center-in-sichuan

[8] http://www.senato.it/

[9] D'Angelo, Gabriele, Stefano Ferretti, and Vittorio Ghini. "Multi-level simulation of internet of things on smart territories." Simulation Modelling Practice and Theory 73 (2017): 3-21.

[10] D'Angelo, Gabriele, Stefano Ferretti, and Vittorio Ghini. "Distributed hybrid simulation of the Internet of things and smart territories." Concurrency and Computation: Practice and Experience 30.9 (2018): e4370.

[11] Velde, Fran. "Bitcoin: A primer." (2013).

[12] Karame, Ghassan, Elli Androulaki, and Srdjan Capkun. "Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin." IACR Cryptology ePrint Archive 2012.248 (2012).

[13] D'Angelo, Gabriele, and Stefano Ferretti. "Highly intensive data dissemination in complex networks." Journal of Parallel and Distributed Computing 99 (2017): 28-50.

[14] Chohan, Usman W. "Assessing the Differences in Bitcoin Other Cryptocurrency Legality Across National Jurisdictions." Available at SSRN 3042248 (2017).

[15] Narayanan, Arvind, et al. Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press, 2016.

[16] Raval, Siraj. Decentralized applications: harnessing Bitcoin's blockchain technology. " O'Reilly Media, Inc.", 2016.

[17] Nakamoto, Satoshi. "Bitcoin: a peer-to-peer electronic cash system (2008)." (2008).

[18] Becker, Georg. "Merkle signature schemes, merkle trees and their cryptanalysis." Ruhr-University Bochum, Tech. Rep (2008).

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

[20] https://bitshares.org/technology/delegated-proof-of-stake-consensus/

[21] https://www.binance.vision/blockchain/proof-of-authority-explained

[22] Naik, Rahul P., and Nicolas T. Courtois. "Optimising the SHA256 Hashing Algorithm for Faster and More Efficient Bitcoin Mining." MSc Information Security Department of Computer Science UCL (2013): 1-65.

[23] Lamport, Leslie, Robert Shostak, and Marshall Pease. "The Byzantine generals problem." Concurrency: the Works of Leslie Lamport. 2019. 203-226.

[24] https://learnmeabitcoin.com/

[25] https://en.bitcoin.it/wiki/Script

[26] Clack, Christopher D., Vikram A. Bakshi, and Lee Braine. "Smart contract templates: foundations, design landscape and research directions." arXiv preprint arXiv:1608.00771 (2016).

[27] https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027

[28] Bai, Qianlan, et al. "A deep dive into blockchain selfish mining." ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019.

[29] Kuikka, Oona. "Can cryptocurrency come to fulfil the functions of money? An evaluation of cryptocurrency as a global currency." (2019).