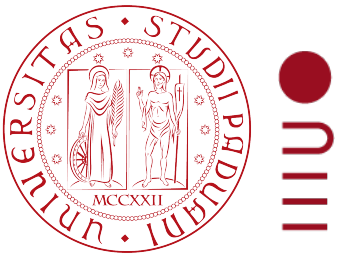

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in
Ingegneria dell'Informazione

Modelli di Code per la Descrizione del Processo di Mining di una Criptovaluta

Relatore:
Prof. Nicola Laurenti

Laureando:
Alberto Molon
1136509

Anno Accademico 2018/2019

Sommario

Negli ultimi anni c'è stato uno sviluppo della tecnologia blockchain, che ha portato molti sviluppatori a creare nuove monete virtuali. Il processo di mining è il processo più importante che sta alla base del funzionamento di una criptovaluta, tramite il quale si creano nuove criptomonete e si confermano blocchi di transazioni. In questa tesi si studiano i comportamenti statistici dei tempi di tale processo utilizzando la teoria delle code. Si approfondiranno due modelli di code e la trattazione si baserà anche sugli articoli presenti in letteratura. Lo studio della statistica dei tempi e degli aspetti che la influenzano, sono elementi importanti per capire quali sono i punti di forza e debolezza dei vari modelli. I risultati ottenuti, mostrano che si ottengono tempi bassi se la transazione ha una commissione alta o un pagamento elevato.

Indice

1. Introduzione

In questi anni, la tecnologia blockchain sta avendo grande considerazione da parte di aziende pubbliche e private per lo sviluppo di reti decentralizzate. Questa tecnologia è conosciuta soprattutto per essere alla base del funzionamento delle criptovalute. I nodi della rete si possono scambiare transazioni in modo sicuro grazie al lavoro di alcuni utenti, chiamati minatori, che attuano il processo di mining per confermare le transazioni e creare nuova valuta.
In questa tesi, viene studiato il comportamento statistico dei tempi di conferma delle transazioni, sfruttando due modelli di teoria delle code e confrontando i risultati presenti in letteratura. Lo studio della statistica dei tempi e degli aspetti che li influenzano sono elementi importanti per capire quali sono i punti di forza e debolezza dei vari modelli.

1.1 Stato dell'arte

Gli articoli presenti in letteratura, utilizzano modelli di code diversi adoperando approcci vari. Ad esempio, in [12], si sfrutta il modello M/M/1 per il calcolo del tempo totale di sistema, visto come somma tra il tempo di mining e il tempo del consenso. Questo articolo, si basa a sua volta su [8], che è interessante perchè spiega dettagliatamente le ipotesi dei processi e fornisce una vasta panoramica dello stato dell'arte. Dopodichè, in [11] e in [7], vengono studiati i fattori che impattano sul tempo di conferma di una transazione. I modelli utilizzati sono: M/G/1 per [11] e M/G^B/1 per [7].
Inoltre, gli articoli [10], [5], [9] e il libro [1], sono stati d'aiuto per la parte riguardante il funzionamento della blockchain e delle criptovalute.

1.2 Organizzazione del testo

La tesi è composta come di seguito:

Il secondo capitolo descrive le conoscenze preliminari per la comprensione del testo.

Il terzo capitolo approfondisce l'analisi del modello M/G/1.

Il quarto capitolo approfondisce l'analisi del modello M/M/1.

Il quinto capitolo illustra le conclusioni.

2. Conoscenze preliminari

Si danno per assunte tutte le conoscenze relative alla teoria delle code viste nel corso di Telecomunicazioni. Per ulteriori dettagli, si veda [2, cap. 8]. Per una facile comprensione, vengono riportate di seguito delle nozioni riguardanti la blockchain, le criptovalute e le catene di Markov.

2.1 Blockchain e criptovalute

La ***blockchain*** è un registro pubblico, condiviso e decentralizzato che non permette la falsificazione delle informazioni registrate (chiamate anche *transazioni* o *entry*). Queste informazioni, sono raggruppate in *blocchi*, collegati tra loro in ordine cronologico. L'insieme di tutti questi blocchi, forma l'intero registro (ledger). La rete blockchain viene realizzata configurando un *sistema peer-to-peer*. Essa è un tipo di rete in cui i nodi (o peer) possono comunicare e condividere risorse tra loro, senza dover passare per un'autorità centrale (come un server). Quest'ultima proprietà, è detta *decentralizzazione*.
Una ***criptovaluta*** è un sistema di scambio digitale peer-to-peer in cui vengono utilizzate tecniche crittografiche per generare e spedire unità di valuta [5]. Dunque, essa implementa un sistema blockchain, in cui i dati registrati sono transazioni finanziarie. In parole povere, una criptovaluta è una moneta digitale che ha un valore di mercato come una moneta fisica. Le criptovalute più conosciute e utilizzate sono: *Bitcoin*, *Ethereum* e *Monero*.

2.1.1 Transazioni

In una blockchain di criptovalute, le *transazioni* sono quelle informazioni pubbliche associate al trasferimento di criptovalute tra partecipanti nel sistema. Ogni transazione non è collegata alla sua precedente in ordine cronologico, ma alla sua *transazione-input*, ovvero al precedente scambio, o ai precedenti scambi, che hanno fornito delle criptovalute al ricevente così da poter diventare ora il mittente nella transazione in questione [13]. Le transazioni possono essere di diversi tipi: [1]

- *Transazioni con resto*: quando si ha un input e due output. Nel mondo reale, sono l'equivalente dell'acquisto di un bene in cui l'acquirente riceve un resto.

- *Transazioni aggregatrici*: quando si hanno più input ed un solo output. Nel mondo reale, sono l'equivalente dello scambiare una pila di monete per una banconota singola di valore maggiore.

- *Transazioni distribuite*: quando si ha un solo input e più output. Nel mondo reale, sono l'equivalente del pagamento da parte di uno stesso pagatore a più riceventi, in una sola operazione; ad esempio quando un'azienda paga gli stipendi a tutti i suoi dipendenti.

2.1.2 Crittografia alla base di una criptovaluta

Ogni utente della rete possiede:

- un portafoglio (wallet) contenente chiavi crittografiche private, generate di volta in volta quando avviene una transazione.

  – il *wallet* è l'equivalente di un portafoglio materiale, che contiene la coppia di chiavi pubblica e privata e mostra il proprio conto. Però, questo quantitativo di criptovalute, non è fisicamente posseduto nel proprio pc o su un server come se fosse una banca, ma è il

risultato degli input e output registrati nelle transazioni salvate nella blockchain.

  - la *chiave privata* è una stringa alfanumerica segreta, generata e gestista dal software del wallet.

- un *indirizzo pubblico*, calcolato a partire dalla chiave crittografica pubblica tramite una funzione di hashing. Questo indirizzo serve per ricevere le transazioni. Gli indirizzi iniziano con i caratteri 1 o 3.

  - la *chiave pubblica* è una stringa alfanumerica visibile a tutti, generata e gestista dal software del wallet. Essa viene ricavata matematicamente dalla chiave privata.

  - la *funzione di hashing* è una funzione matematica che non è invertibile e mappa dati di qualsiasi lunghezza in stringhe di bit di lunghezza fissa (chiamate *hash* o *impronte*). Le funzioni di hash inoltre sono veloci da calcolare e resistenti alle collisioni perchè se si cambia anche un solo bit in ingresso, l'output viene stravolto (dunque è garantita l'integrità dei dati). Inoltre, essendo una funzione non invertibile, è altamente improbabile ottenere il dato di ingresso avendo a disposizione il suo hash.

Le chiavi crittografiche sono importanti perchè sono alla base della *firma digitale*, lo strumento che garantisce sia l'autenticazione del mittente sia l'integrità dei dati. La firma digitale su un documento consiste nel cifrare il documento tramite la chiave privata e utilizzare la relativa chiave pubblica, che è nota a tutti, per decifrare la firma e quindi constatare o meno l'identità del mittente. Ogni proprietario trasferisce le criptovalute firmando digitalmente l'hash della transazione precedente (quella che viene chiamata transazione-input) e la chiave pubblica del futuro proprietario, e aggiungendo queste informazioni alla transazione corrente [9]. In questo modo, le transazioni sono legate tra loro e quando un utente vuole spendere le

sue criptovalute (quindi fare una nuova transazione), dovrà "sbloccare" la catena fornendo la sua chiave pubblica. Attraverso la presentazione della chiave pubblica e della firma, tutti i membri della rete possono decifrare la firma con la chiave pubblica del mittente e verificare che l'hash risultante corrisponde all'impronta digitale di quella transazione. Di conseguenza:

- la transazione è valida e integra (la firma è infatti legata all'impronta digitale e quindi alla transazione stessa. Una modifica della transazione produrrebbe un'impronta digitale totalmente diversa);
- l'identità del mittente è autentica.

Questo processo di verifica e autenticazione è mostrato nella fig. 2.1

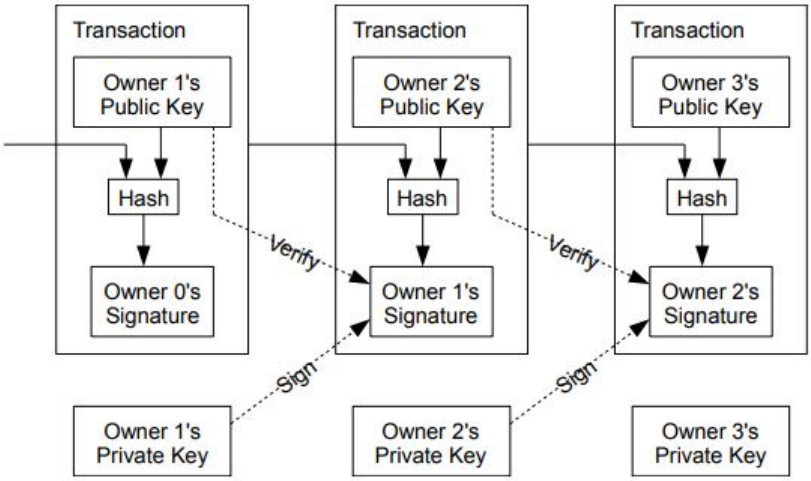

Figura 2.1: Autenticazione della catena di transazioni

2.1.3 Struttura dei blocchi

Come dice il nome, la blockchain è una catena di blocchi, contenenti transazioni valide. L'intera rete è formata da utenti chiamati *nodi*, che possono essere localizzati in qualsiasi parte del pianeta. Ciascun nodo contiene il registro di tutte le transazioni effettuate nella rete fino a quel momento. Un qualsiasi blocco contiene le seguenti informazioni [5]:

- la dimensione (in byte) del blocco;

Campo	Descrizione
Previous Block Hash	Hash del blocco precedente (parent block) a cui è collegato
Merkel Root	Hash della radice del merkel tree associato al blocco
Timestamp	Ora di creazione del blocco
Difficulty Target	Target usato per il PoW
Nonce	Numero contatore usato per il PoW

Tabella 2.1: Struttura dell'intestazione

Campo	Descrizione
Block Size	Dimensione del blocco in bytes
Block Header	Intestazione del blocco contenente diversi campi
Counter	Numero totale di transazioni registrate
Transactions	Transazione registrate nel blocco

Tabella 2.2: Struttura di un blocco

- l'*intestazione* (header) con diversi campi al suo interno, mostrati nella tab. 2.1;

- il numero totale di transazioni. Ogni blocco è identificato in maniera univoca da un *hash*, generato usando l'algoritmo crittografico di hashing sull'intestazione del blocco (per il Bitcoin, per esempio, l'algoritmo è SHA-256). La tab. 2.2 mostra la struttura di un blocco.

I blocchi sono collegati "all'indietro" (back-linked), cioè ognuno si riferisce al blocco precedente presente nella catena. Per questo motivo, la blockchain è spesso visualizzata come una pila verticale, con blocchi stratificati l'uno sopra l'altro. Questa visualizzazione, provoca l'uso di terminologie come *altezza* (height) per riferirsi alla distanza dal primo blocco e *top* o *tip* (cima o punta) per riferirsi al blocco aggiunto più recentemente. Ciascun blocco inoltre referenzia il blocco precedente, conosciuto anche come *parent block* (blocco genitore), attraverso il campo *previous block hash* dell'intestazione. La sequenza di hash che collegano ogni blocco al proprio genitore crea una catena che si collega, blocco per blocco fino al primo blocco creato, denominato *genesis block*. Una semplice rappresentazione è mostrata nella fig. 2.2. Si supponga che un genitore venga modificato in qualsiasi

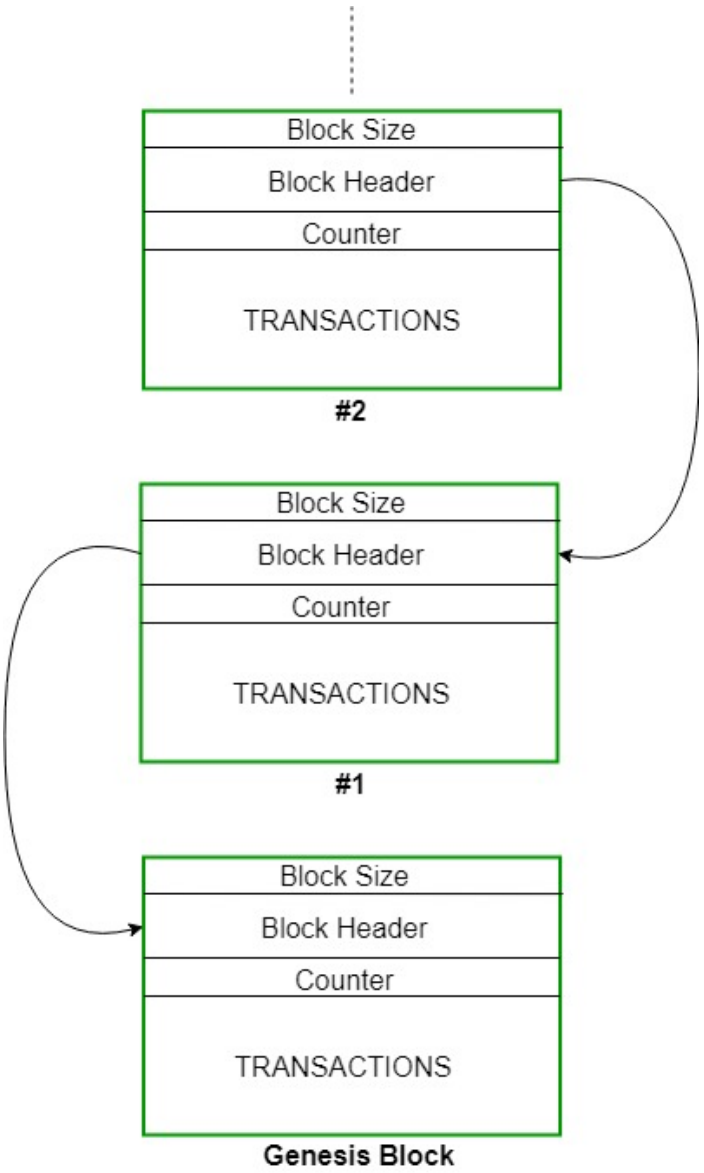

Figura 2.2: Blockchain vista come una catena

modo. Allora il suo hash cambia. L'hash modificato necessita un cambio nel puntatore *previous block hash* del figlio. Questo, a sua volta, causa il cambio dell'hash del figlio, che richiede il cambio nel puntatore dell'hash nipote, che a sua volta cambia il puntamento al nipote, e così via. Questo effetto a cascata assicura che, fino a quando un blocco ha tante generazioni di blocchi che lo seguono, non può essere modificato senza forzare un ricalcolo su tutti i blocchi seguenti. Visto che per questo ricalcolo servirebbe un'enorme potenza computazionale, l'esistenza di una lunga catena di blocchi fa sì che la storia più profonda della blockchain sia immutabile, che è una degli elementi chiave della sicurezza della blockchain [1].

2.1.4 Funzionamento

Le nuove transazioni devono per prima cose essere confermate, cioè bisogna verificare la loro firma digitale. Questo procedimento, lo possono fare tutti gli utenti della rete. La firma digitale, però, non offre protezione contro il *double spending*, cioè quando un utente spende più volte delle criptovalute già spese [9]. Per risolvere questo problema, bisogna tener conto anche dei vecchi blocchi (e quindi delle vecchie transazioni), e provare che quelli nuovi, sono legati matematicamente ai precedenti. Siccome questa verifica deve essere rigorosa e non esiste un'autorità centrale, il compito spetta ai nodi della rete (lo possono fare tutti o solo alcuni). Questi nodi specializzati, prendono il nome di *minatori* (o miner) e il processo di verifica che eseguono si chiama *mining*. Esso può essere diverso da criptovaluta a criptovaluta. La conferma avviene trovando il valore di hash che soddisfa una certa proprietà (procedimento chiamato *proof of work*, abbreviato PoW). Quando si trova una soluzione, il nuovo blocco è ritenuto valido e viene aggiunto al registro di ogni nodo. Il risultato finale è una catena di blocchi approvati non modificabili e in ordine cronologico.

2.1.5 Mining di una criptovaluta

Per *mining* si intende quel processo svolto da un insieme di utenti della rete che, fa eseguire all'hardware di ogni utente calcoli matematici al fine di confermare il nuovo blocco ed aumentare la sicurezza della rete [3]. Questo processo ha due scopi:

- mantiene l'integrità e l'autenticità della blockchain, assicurando che le transazioni siano confermate solo se è stata usata sufficiente potenza di calcolo per il blocco che le contiene;
- emette una quantità stabilita di nuove criptovalute (quasi come una banca centrale emette nuova moneta), che spettano al minatore che per primo le ha prodotte. A tale quantità prestabilita va a sommarsi anche il totale delle commissioni delle transazioni registrate nel blocco [5].

Dunque, la sicurezza della blockchain è resa possibile da quanti più nodi "onesti" sono presenti nella rete, in modo da rendere difficile (se non impossibile) il lavoro dei nodi "disonesti" che vogliono invece modificare il registro a loro vantaggio. L'onestà dei nodi è "comprata" attraverso un particolare sistema di attribuzione di ricompense, che incentivano tale onestà [13]. Quando arrivano delle nuove transazioni, esse vengono aggiunte a un pool temporaneo di transazioni non verificate. Questo pool, è memorizzato da ogni minatore. Man mano che i minatori generano un nuovo blocco, prelevano le transazioni non verificate dal pool. Esse vengono aggiunte al nuovo blocco, ordinate secondo certi criteri (ad esempio, dando priorità a quelle con commissioni maggiori). Da qui in poi, ogni minatore inizia il processo di mining di un nuovo blocco di transazioni. Nel momento in cui riceve un blocco valido da un altro minatore, capisce di aver perso la sfida. Dunque lo controlla (procedimento chiamato *consenso*) e lo aggiunge alla sua blockchain. Dopodichè, ritenta la sfida con un nuovo blocco, creato a partire da altre transazioni e il procedimento si ripete. Più nel dettaglio,

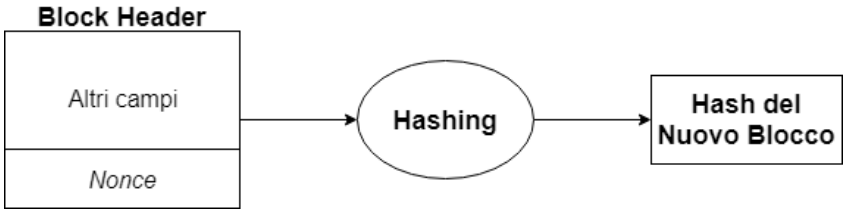

Figura 2.3: Input utilizzato per il calcolo dell'hash del nuovo blocco

il funzionamento del processo di mining è il seguente: il minatore ha come input l'intestazione del blocco nuovo e un numero, chiamato *nonce* (che, tra l'altro, è un campo presente nell'intestazione). Il nonce è un numero (spesso scelto arbitrariamente) che viene utilizzato solo una volta per questo scopo e poi scartato [10]. L'input è rappresentato nella fig 2.3. Ora, partendo da questo input, il minatore calcola il corrispettivo hash e verifica se ha una certa proprietà/struttura definita dalla criptovaluta. Se così non fosse, cambia nonce e ritenta. Il compito del minatore, è dunque quello di trovare il nonce che soddisfa la proprietà. L'insieme di questi compiti matematici, prende il nome di *proof of work*, abbreviato in PoW (prova del lavoro). La complessità del proof of work sta proprio nell'ottenere il nonce corretto, in quanto questo comporta il calcolo di un numero esponenziale di hash [5]. Una volta che un minatore trova la soluzione per validare il blocco, manda a tutti gli altri minatori della rete:

- il nuovo blocco appena confermato;
- il nonce corretto;
- l'hash del nuovo blocco (calcolato con il nonce corretto), che è legato all'ultimo blocco, in quanto, per calcolarlo, è stato utilizzato il suo hash.

Essi, dovranno confermare che il risultato trovato è corretto. Se è così, il nuovo blocco viene aggiunto "in cima" alla blockchain e il minatore che ha validato il blocco per primo otterrà la ricompensa. Tale verifica, chiamata

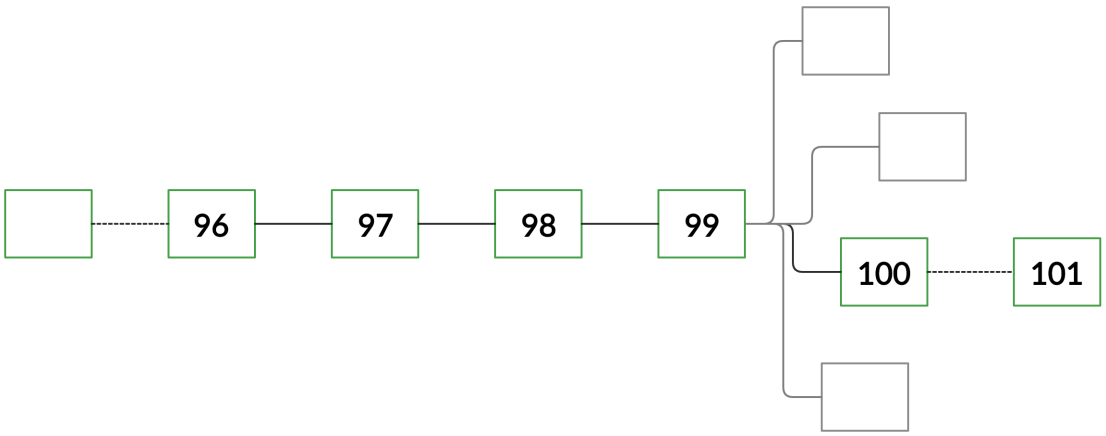

Figura 2.4: Esempio di blockchain ramificata

anche *consenso*, è semplice e veloce, in quanto basta calcolare un solo hash e verificare che combaci [5].

2.1.6 Numero di conferme

Nella realtà, succede molto spesso che i minatori scelgono transazioni diverse per un blocco (scegliendo, per esempio, quelle con commissioni elevate). Quindi ognuno può lavorare su un potenziale blocco diverso che cercherà di confermare prima degli altri. Quello che succede in questo caso, è che si formino delle *ramificazioni* nella parte finale della blockchain. Quando avvengono queste situazioni, la soluzione da adottare è specificata negli algoritmi delle criptovalute. Una regola molto comune, è quella di far continuare la blockchain sul suo ramo più lungo, mentre tutto il resto viene ignorato. Nella fig. 2.4 è mostrato un esempio di blockchain ramificata, che continua sul ramo più lungo. Basandoci sulla fig. 2.4, supponiamo che l'ultimo blocco confermato della blockchain sia il blocco numero 100. Ora, supponiamo che un minatore malintenzionato voglia contraffare questo blocco e quindi lo modifichi. Ovviamente, potrà apporre tale modifica solo sulla copia della blockchain che ha sul suo server. Dopodichè, dovrà far accettare (e quindi divulgare) la sua copia "contraffatta" al resto della rete. Per riuscire nel suo intento, il malintenzionato dovrà:

- trovare un nonce per il blocco 100 che lui ha modificato, in modo che il sistema non evidenzi il suo tentativo di contraffazione;
- aggiudicarsi la gara sul blocco 101 (perchè si continua sul ramo più lungo).

In sostanza, dovrà riuscire a trovare 2 nonce (per il blocco 100 e blocco 101) prima che un qualunque altro minatore riesca a trovarne uno solo, ovvero quello per il blocco 101. Il tutto sarà esponenzialmente più improbabile se vorrà contraffare una transazione del blocco 99 perchè dovrà trovare tre nonce... e così via per i blocchi precedenti. Per questi motivi, per misurare la sicurezza di una transazione, si indicano il numero delle conferme che ha ricevuto: ad esempio, se l'ultimo blocco confermato è il numero 100, allora una transazione presente nel blocco 98 si dice abbia 3 conferme, ovvero la conferma del blocco 98, quella del 99 e quella del 100.

2.1.7 Esempio: Mining del Bitcoin

Il processo di mining di un Bitcoin si basa sull'algoritmo *Hashcash* [9]. Il funzionamento è molto simile a quello descritto in precedenza per una criptovaluta generale. In particolare, Bitcoin utilizza *SHA-256* come funzione di hash (ciò significa che l'impronta generata sarà di 256 bit). Inoltre, il minatore deve trovare quel nonce tale per cui l'hash di output inizi con n_z zeri, dove n_z sono i bit più significativi [5]. Per vedere se il nonce utilizzato è quello corretto, il minatore controlla se l'hash risultante è minore o uguale di un certo valore di 256 bit, chiamato *target* (che tutti i minatori conoscono). Se è più piccolo, significa che l'output aveva almeno uno zero inziale in più rispetto al target e dunque il minatore ha trovato la soluzione; altrimenti, il nonce viene incrementato di uno e si riprova (il nonce parte da zero). Il valore target viene ricalcolato ogni 2016 blocchi (circa due settimane). Ad esempio, se il target corrente fosse di 00000111 e l'hash di output 001110010, il minatore non avrebbe trovato una soluzione

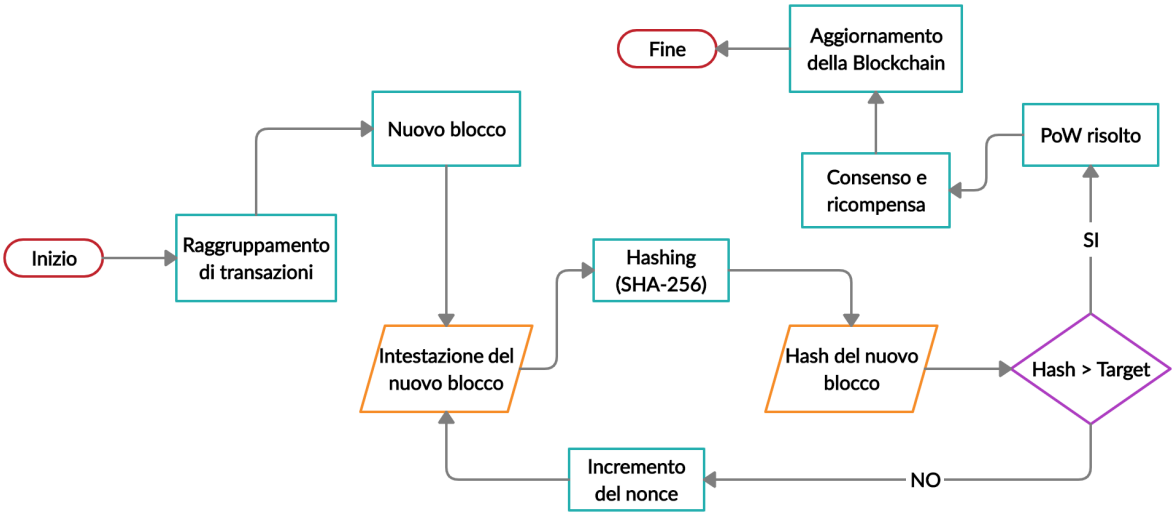

Figura 2.5: Diagramma a blocchi del processo di mining del Bitcoin

valida, perchè il target ha 5 zeri iniziali e il risultato solo 2; il minatore deve aumentare progressivamente il nonce ad ogni nuovo tentativo, finchè il risultato non inizia con almeno 6 zeri [13]. La fig 2.5 mostra il diagramma a blocchi del processo di mining del Bitcoin. Il lavoro medio richiesto è esponenziale rispetto al numero di zeri. In più, SHA-256 richiede un *hash rate*[1] elevato, di giga hashes al secondo (GH/s) o superiore [10]. Il tempo medio necessario per estrarre un blocco Bitcoin con SHA-256 è circa di 10 minuti [3]. Il procedimento di proof of work del Bitcoin è identico a quello descritto precedentemente per una criptovaluta generale. Il minatore che ha risolto per primo il blocco, verrà premiato con dei Bitcoin. Il premio varia in base al numero di Bitcoin estratti. Poichè c'è nè un numero limitato, i premi diminuiscono della metà dopo ogni 210.000 blocchi estratti (circa ogni 4 anni). A partire da agosto 2018, i minatori vengono premiati con 12,5 Bitcoin. Nei primi anni, cioè nel 2009, i minatori guadagnavano 50 Bitcoin [5]. Oltre alla ricompensa ottenuta dal mining, i minatori ricevono anche un importo chiamato *commissione di transazione* per ogni transazione confermata che aggiungono alla blockchain [10]. Per convenzione,

[1]Per *hash rate* si intende l'unità di misura della potenza di elaborazione della rete di una criptovaluta, cioè quanti calcoli matematici riesce ad eseguire in un certo tempo. Ad esempio, quando la rete raggiunge un hash rate di 10 TH/s, significa che può realizzare un trilione di calcoli al secondo. [3]

una transazione che riceve 6 conferme (cioè sopra il suo blocco ne vengono aggiunti altri 5) viene considerata sicura al 100%.

2.2 Catena di Markov

Una ***catena di Markov*** è un processo stocastico avente una proprietà caratterizzante, chiamata *proprietà di Markov* [2]. Il processo, che può essere a tempo continuo o a tempo discreto, è determinato da una successione di variabili aleatorie $\{X_w\}$, che prendono valori in un insieme S, detto *spazio degli stati*. Esso può essere finito o al più infinito numerabile. L'indice w di $\{X_w\}$ indica il *tempo*, mentre i possibili valori della successione prendono il nome di *stati* del processo. Al trascorrere del tempo, il processo può "saltare" da uno stato all'altro. Se ad un certo istante w si trova in uno stato i ed all'istante successivo ($w+1$ se è discreto oppure $w+u$ se è continuo) si trova in uno stato $j \neq i$, si dirà che c'è stata una *transizione*. Siccome i processi coinvolti sono stocastici, i calcoli coinvolgeranno le probabilità. Le più importanti sono le cosiddette *probabilità di transizione*. Per il *tempo discreto* valgono:

$$P_{i,j}(k) = P[X_{n+k} = j | X_n = i] \tag{2.1}$$

che indica la probabilità di trovarsi nello stato j al tempo $n+k$ sapendo di essere nello stato i al tempo n. In questo caso, k indica il *numero di step* per andare dallo stato i allo stato j.
Per il *tempo continuo* invece, valgono:

$$P_{i,j}(u) = P[X(t_0 + u) = j | X(t_0) = i] \tag{2.2}$$

che indica la probabilità di trovarsi nello stato j al tempo $t_0 + u$ sapendo di essere nello stato i al tempo t_0. In questo caso, u indica l'*intervallo temporale* per andare dallo stato i allo stato j.

Si dirà che queste probabilità sono *stazionarie* se dipendono solo dalla lunghezza temporale k o h (a seconda che il tempo sia discreto o continuo) e non dai singoli istanti n o t_0. Le relative catene di Markov, invece, si diranno *omogenee*. I processi che si andranno a sviluppare nei capitoli successivi, sono tutti di questo tipo.

2.2.1 Proprietà di Markov

La proprietà di Markov per un processo a tempo discreto è sintetizzata nella seguente formula:

$$P[X_{n_k} = \sigma_k | X_{n_{k-1}} = \sigma_{k-1}, \dots, X_{n_0} = \sigma_0] = P[X_{n_k} = \sigma_k | X_{n_{k-1}} = \sigma_{k-1}]. \tag{2.3}$$

Mentre, per un processo a tempo continuo, vale:

$$P[X_{t_n} = \sigma_n | X_{t_{n-1}} = \sigma_{n-1}, \dots, X_{t_0} = \sigma_0] = P[X_{t_n} = \sigma_n | X_{t_{n-1}} = \sigma_{n-1}], \tag{2.4}$$

dove i σ_i sono elementi appartenenti allo spazio di stato $S = \{0, 1, 2, \dots\}$. L'interpretazione della proprietà di Markov è la seguente. Consideriamo:

- l'istante n_{k-1} (o nel continuo t_{n-1}) come il "presente" del processo;
- gli istanti $n_0, n_1, \dots, n_{k-2}$ (o nel continuo $t_0, t_1, \dots, t_{n-2}$) come il "passato" del processo;
- l'istante n_k (o nel continuo t_n) come il "futuro" del processo.

Dunque, le equazioni (2.3) e (2.4) ci dicono dice che la conoscenza del "passato" del processo non predice la sua evoluzione futura dallo stato σ_{k-1} in cui si trova al momento. Perciò, è possibile "dimenticarsi" della storia passata, conoscendo esclusivamente lo stato presente.

3. Analisi del modello M/G/1

Il modello M/G/1 simboleggia un sistema che ha un solo servitore, in cui gli arrivi sono markoviani (o senza memoria), mentre il servizio segue una distribuzione generica $F_y(a)$. Per fissare le idee, diciamo che il tasso degli arrivi è pari a λ, mentre il tasso di servizio è pari a μ. La markovianità degli arrivi è giustificata dal fatto che in una rete grande, le possibili sorgenti di transazione sono molte e indipendenti; mentre la stazionarietà/omogeneità è giustificata dal fatto che il comportamento della rete è tempo invariante. La statistica dei tempi di inter-arrivo è esponenziale perchè è conseguenza della poissonianità e omogeneità. Il fatto che il sistema abbia un solo servitore, non significa che anche il numero di minatori dell'intera rete sia uno, ma significa che la stessa transazione (che è il cliente del sistema) viene servita da tutti gli m minatori. Dunque, in termini di teoria delle code, è come avere un sistema formato da una sola coda in cui il cliente viene servito da un'unica entità composta da m minatori che la servono contemporaneamente. Il modello M/G/1 è utilizzato in letteratura per il calcolo dei tempi di conferma delle transazioni e dei ritardi e dello studio dei fattori che li influenzano, come ad esempio la dimensione del blocco o il numero di transazioni.

3.1 Rappresentazione come catena di Markov

Il modello M/G/1 si può analizzare facilmente se lo si rappresenta come una catena di Markov. Per farlo, però, bisogna dimostrare la prorpietà di Markov e trovare lo stato e la sua statistica. Fatto questo, poi, si possono ricavare le statistiche e i momenti di tutte le metriche. Si inizia considerando il numero totale di clienti $x(t)$ nel sistema negli istanti immediatamente

successivi alla n-esima partenza dal sistema, denominati t_n^+.[1] Indicando con $x_n := x(t_n^+)$ il numero di clienti nel sistema immediatamente dopo il verificarsi della n-esima partenza, possiamo scrivere le seguenti relazioni che ne specificano la dinamica [2]:

- se la partenza n-esima lascia il sistema vuoto, prima che si verifichi una nuova partenza, è necessario attendere l'arrivo di un cliente e la fine del suo tempo di servizio. Supponiamo che in questo lasso di tempo, arrivi un numero di clienti pari alla variabile aleatoria v_n. Dunque, un instante dopo la partenza n-esima, il numero totale di clienti diventa pari a v_n. In formula:

$$x_{n+1} = v_n \tag{3.1}$$

- se dopo la partenza n-esima il sistema non è vuoto, il prossimo cliente in coda accede al servizio e si aggiungono poi v_n clienti al sistema. Dunque, dopo la partenza, avremo $x_n - 1$ clienti rimanenti e in più i v_n appena arrivati. In formula:

$$x_{n+1} = x_n - 1 + v_n \tag{3.2}$$

  Possiamo unificare le due equazioni precedenti utilizzando la funzione indicatrice:

$$\chi\{x_n > 0\} = \begin{cases} 1 & \text{se} \quad x_n > 0 \\ 0 & \text{se} \quad x_n = 0 \end{cases} \tag{3.3}$$

  ottenendo dunque:

$$x_{n+1} = x_n + v_n - \chi\{x_n > 0\}. \tag{3.4}$$

Ora, per poter dire che il sistema M/G/1 evolve secondo una catena di Markov, bisogna dimostrare che la descrizione statistica fatta per gli istanti t_n^+, valga *per ogni* istante di osservazione. Un modo per dimostrarlo è

[1] Il termine *clienti* è usato per questo modello in maniera generale. Infatti, ci si può riferire ai clienti sia come transazioni sia come blocchi.

il seguente. Si può notare che, asintoticamente, il numero x_n di clienti nel sistema subito dopo una partenza, è distribuito come $x(\tau_m^-)$, che è il numero di clienti nel sistema appena prima l'arrivo di un nuovo cliente. Dopodichè, si può sfruttare la proprietà PASTA, che dice che gli arrivi poissoniani "vedono" davanti a loro, la distribuzione asintotica dello stato $x(t)$. Dunque, $x(\tau_m^-)$ è distribuita come $x(t)$ per $t \to \infty$. Ma allora, se asintoticamente x_n è distribuita come $x(\tau_m^-)$ che, a sua volta, è distribuita come $x(t)$, vale che per $n \to \infty$, x_n converge alla distribuzione di $x(t)$. In conclusione, in regime stazionario e in qualsiasi istante di osservazione, si ha un'unica descrizione statistica dello stato [2].

3.2 Calcolo del tempo medio di attesa in coda

Ora, si vuole riportare il calcolo del tempo medio di attesa in coda $E[T_Q]$, che in ambito criptovalute, viene visto come il *ritardo* del tempo di conferma di una transazione. Per tutti gli altri momenti delle metriche, si consiglia di vedere [2, pp. 586–587]. Viene sfruttata la tecnica *tagged-job*, nella quale si "tagga" un arrivo arbitrario e si ragiona sul tempo speso in coda proprio da questo arrivo selezionato [6]. Si ipotizza che tutte le variabili in gioco siano i.i.d. Verrà utilizzata la seguente notazione:

- $\rho = \frac{\lambda}{\mu}$ = fattore di carico;
- T_Q = tempo in coda;
- N_Q = numero clienti in coda;
- N_Q^A = numero clienti in coda visti dal cliente "taggato";
- S = tempo di servizio $\implies E[S] = \frac{1}{\mu}$;
- S_i = tempo di servizio i-esimo cliente;
- S_e = tempo di servizio visto dal cliente "taggato".

Allora:

$$
\begin{aligned}
E[T_Q] &= E[\text{tempo di servizio dei clienti in coda prima del “tag”}] \\
&\quad + E[\text{tempo di servizio residuo del cliente in servizio}] = \\
&= E[\sum_{i=1}^{N_Q^A} S_i] + E[\text{tempo di servizio residuo del cliente in servizio}] = \\
&\underset{\text{indip.}}{=} E[N_Q^A]E[S] + P[\text{unico servitore è impegnato}]E[S_e] = \\
&\underset{\text{PASTA}}{=} E[N_Q]E[S] + \rho E[S_e] = \qquad (3.5a) \\
&\underset{\text{Little}}{=} E[T_Q]\lambda E[S] + \rho E[S_e] = \\
&= \rho E[T_Q] + \rho E[S_e] = \\
&= \frac{\rho}{1-\rho} E[S_e]. \qquad (3.5b)
\end{aligned}
$$

3.3 Analisi della statistica del ritardo

Nell’articolo [11], si parla dei fattori che impattano i ritardi delle transazioni Bitcoin sfruttando la teoria delle code. Per *ritardo* si intende quell’intervallo temporale che la transazione deve attendere prima di essere scelta per far parte di un blocco. Utilizzando la nomenclatura della teoria delle code e considerando le transazioni come clienti, il ritardo equivale al tempo medio di attesa in coda, che in questo caso viene chiamata in gergo *mining pool*. In [11, sez. 4] sono scritti i parametri considerati. Vengono riportati quelli più rilevanti:

- λ è il tasso di arrivo di un gruppo di transazioni;
- λ_B è il tasso con cui vengono minati i blocchi (misurato in $blocchi/s$);
- ogni blocco contiene in media τ transazioni;
- $\lambda = \lambda_B \tau$ è il tasso di servizio delle transazioni;

- B è il tempo che intercorre tra la conferma di due blocchi (tempo inter-blocco). Inoltre vale che $E[B] = \frac{1}{\lambda_B}$;
- D è il ritardo di una transazione. In particolare, $D_{i,j}$ indica il tempo dall' istante della conferma della j-esima transazione nel blocco i fino all'istante di conferma dell'intero blocco.

Dopodichè, sfruttando la tecnica *tagged-job*, gli autori hanno affermato che:

$$E[D] = \alpha E[B] + E[B_r], \tag{3.6}$$

dove:

- $E[B_r]$ è il tempo medio residuo del blocco in servizio;
- α denota il numero medio di blocchi che un utente deve attendere prima che la sua transazione sia confermata. Una transazione è confermata quando il suo blocco è aggiunto alla blockchain e poi vengono aggiunti altri blocchi sopra di esso (blockchain come pila, fig. 2.2 della sottosez. 2.1). Questo parametro, viene poi calcolato tramite simulazioni.

Infatti, la formula (3.6) dice che il "tag", deve tener conto del tempo medio residuo del blocco davanti a lui, ma anche del tempo medio che un altro blocco si aggiunga sopra al suo nella blockchain, perchè così conferma il suo blocco. Si può notare che la formula (3.6) è molto simile alla formula (3.5a) ricavata sopra per un sistema M/G/1. Dopodichè, in [11, sez. 5], gli autori hanno condotto tre simulazioni, scegliendo altrettanti intervalli temporali e usando le statistiche del sito Bitcoin. In tutte e tre le prove, si può notare che il numero delle transazioni con commissioni medie elevate, è maggiore rispetto al numero di quelle con commissioni basse e dunque, le prime vengono confermate in un tempo più corto (in media 30 minuti). Questo risultato è scontato, perchè i minatori cercano sempre il maggior profitto (il tutto è riportato nella tab. 3 [11, sottosez. 5.1]). Inoltre, gli autori hanno

trovato tre valori del parametro α, il cui risultato è $\alpha < 1$. La conclusione di ciò, è che per avere una conferma attendibile della transizione, basta che il blocco che la contiene, venga inserito nella blockchain (non serve aspettare l'aggiunta di altri blocchi).

3.4 Analisi della statistica dei tempi di conferma

Nell'articolo [7], invece, viene sviluppato un modello per il Bitcoin, nominato M/G^B/1, la cui particolarità è il *batch service*. Un sistema di questo tipo, funziona esattamente come il tradizionale M/G/1 in cui però, se al termine di un servizio sono presenti clienti in coda, essi vengono serviti in gruppo (batch). La dimensione del batch è limitata e costante. In questo caso specifico, i clienti sono le transazioni, che vengono processate in blocco quando il minatore è libero. Gli autori di [7], inoltre, assumono che se le transazioni arrivano nel sistema quando è già in atto un processo di mining, esse vengono aggiunte al blocco in servizio solo se la sua dimensione è più piccola della massima dimensione possibile. Viene indicato con b il numero massimo di transazioni che può contenere un blocco. Il calcolo di b è stato fatto studiando le statistiche su vari intervalli temporali. Bitcoin usa come dimensione di blocco circa 1 MByte e all'interno si trovano in media 1750,27 transazioni (per maggiori dettagli si veda [7, sottosez. 3.2]). Nel modello di [7], è usato $b = 2000$ per riferisi ad un blocco di 1 MByte. In [7], gli autori studiano il tempo medio di conferma delle transazioni (indicato con $E[T]$ o TCT) considerando anche le priorità che esse hanno, a differenza di [11], dove quest'ultima non veniva presa in considerazione. La priorità può essere di qualsiasi tipo. Spesso, si dice che una transazione ha un'alta priorità se ha una commissione elevata. Nell'articolo, invece, viene data alta priorità se la transazione ha un alto importo. Inoltre, è riportato che le transazioni che hanno un maggiore pagamento sono quelle che vengono scelte per prime dai minatori e quindi, il loro tempo di con-

ferma sarà più piccolo. Lo studio è stato condotto tramite simulazioni, dividendo le transazioni in due classi: la *classe H* per quelle che hanno un valore maggiore di 1 BTC e la *classe L* per quelle il cui valore è minore. Il motivo della scelta di questo valore per la divisione in classi non è illustrata nell'articolo. Inoltre, in [7, sottosez. 4.2], la *classe L* viene divisa ancora in altre due classi, *L1* ($importo < 0.01\ BTC$) e *L2* ($importo \geq 0.01\ BTC$). In [7, fig. 1] è mostrata la percentuale di transazioni (con relativa classe) presente nel blocco in un arco temporale di due anni. La figura è poco chiara, perchè non si vede il contributo della *classe L1*. Nemmeno in questa sottosezione vengono spiegati i motivi delle scelte dei valori BTC per la divisione in classi. Ritengo che la divisione in *L1* e *L2* serva per far vedere (tramite la fig.1) che negli ultimi anni, ci sono state più transazioni di *L2* che di *L1* (oltre a quelle di *H*). Però, proprio perchè si vogliono analizzare le statistiche dei tempi dei micro-pagamenti, gli autori avrebbero potuto studiare anche *L1* e *L2* oltre che *H* e *L*. Questo articolo, quindi, ci dice che, oltre alle commissioni elevate (studio condotto da [11]), un altro parametro da tener in considerazione per velocizzare i tempi di conferma, è il valore del pagamento che riporta la transazione. In realtà, questi due parametri sono collegati tra loro, perchè si suppone che chi trasferisce grandi quantità di criptomoneta, possa pagare anche delle maggiori commissioni.
In conclusione, [7] mostra i seguenti casi:

- *senza classe*: mostrato in fig. 3.1; il caso più comune è quello di $b = 2000$, in cui il tempo di conferma cresce rapidamente se $\lambda > 3$. Raddoppiando b, il tempo effettivamente diminuisce, ma con $b = 8000$, abbiamo che il sistema diventa instabile se $\lambda > 12$. Ora, in termini di λ non c'è una grande variazione (si passa da 3 a 12), cioè si rimane sullo stesso ordine di grandezza. Questo fa capire che aumentare la dimensione massima del blocco non è un metodo efficace per mitigare considerevolmente il tempo di conferma;

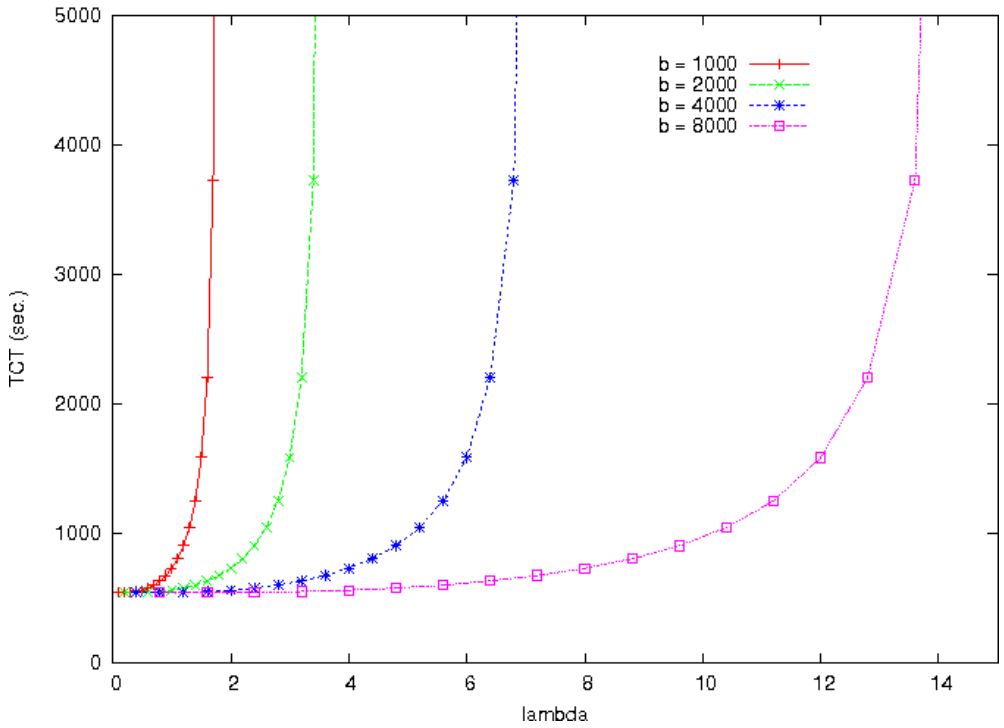

Figura 3.1: Tempo medio di conferma delle transazioni (TCT): caso *senza classe* [7, fig. 5]

- *classe L e H*: mostrato in fig. 3.2; ciascuna classe ha un tasso degli arrivi diverso, rispettivamente λ_L e λ_H. Il grafico mostra il tempo medio in funzione di λ_L tenendo costante λ_H al valore $\lambda_H = 0.2535592656$ (valore calcolato a partire dalle statistiche fornite nel sito Bitcoin). I risultati mostrano quello che ci si aspettava, ovvero il tempo di conferma è minore per le transazioni di classe H. Anche in questo caso, aumentare la dimensione dei blocchi non riduce considerevolmente il problema (fig. 3.2).

3.5 Analisi del modello M/D/1

Il modello M/D/1 è un caso particolare di M/G/1, dove il tempo di servizio è costante: $y = T$ [2]. La statistica del tempo di servizio y è la seguente:

$$p_y(a) = \delta(a - T), \quad m_y = T, \quad \sigma_y^2 = 0. \tag{3.7}$$

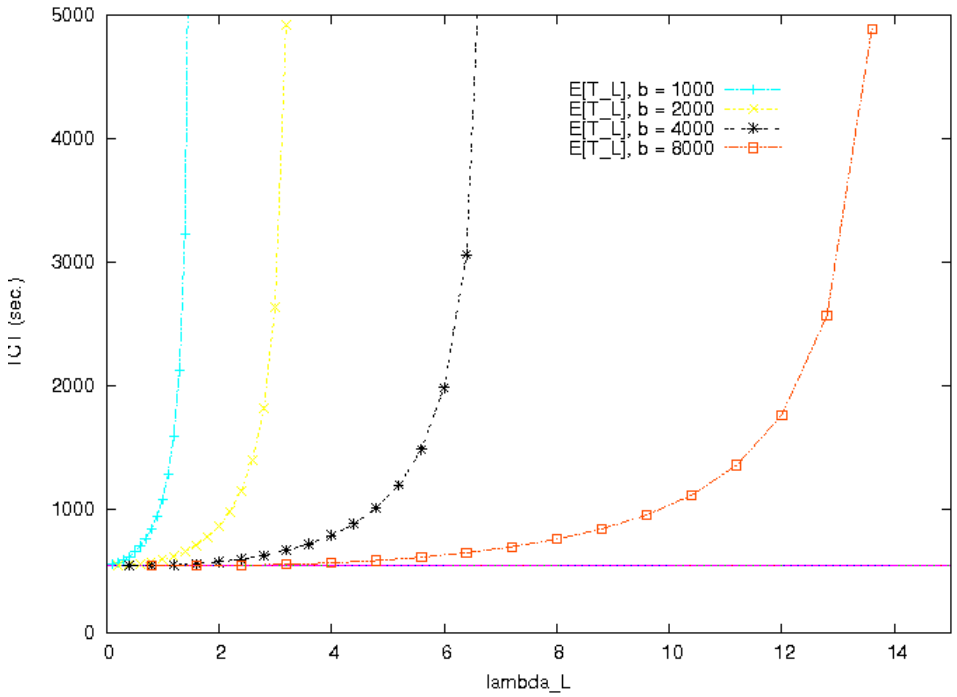

Figura 3.2: Tempo medio di conferma delle transazioni (TCT): caso *classe L e H* [7, fig. 6]

Gli arrivi rimangono un processo poissoniano con tasso λ. Per i calcoli dei momenti delle metriche, si consiglia di vedere [2, p. 590].
La scelta del tempo di servizio è un compromesso tra i seguenti criteri:

- non deve essere troppo grande, perchè in tal caso, un utente malintenzionato che ha appena speso le sue criptovalute, potrebbe in quel tempo sfruttarne la vulnerabilità e fare un double-spending;
- non essere troppo corto, perchè la soluzione del PoW deve essere tale da non poter essere truccata e, se si avesse a disposizione poco tempo, questo rischio potrebbe verificarsi perchè verrebbe scelta una soluzione "banale".

4. Analisi del modello M/M/1

Il modello M/M/1 è un caso particolare di M/G/1, in cui gli arrivi e il servizio sono markoviani (o senza memoria). I tempi inter-arrivo sono variabili aleatorie esponenziali con tasso λ. Stessa cosa dicasi per il servizio, con la differenza che i tempi hanno tasso μ. Per i calcoli dei momenti delle metriche, si consiglia di vedere [2, pp. 546–552].
Nell'articolo [12], si utilizza un sistema M/M/1 per calcolare il tempo totale di conferma di un blocco, che è pari alla somma del tempo di generazione (tempo impiegato dal processo di mining) e il tempo dovuto al consenso e all'aggiunta nel registro. Questo articolo, però, è poco rigoroso nel presentare le ipotesi e i procedimenti matematici che portano ai risultati finali. Infatti, le ipotesi si basano su quelle dell'articolo [8], in cui si fa riferimento al modello G/M^B/1, che ha la particolarità di avere un *batch service*. Questa ipotesi, non viene specificato in [12]. Mentre, per quanto riguarda i procedimenti matematici, dall'eq. 6 in [12, sez. 3], viene utilizzato il parametro m, che viene definito come il *numero di servitori nel sistema*. Inoltre, dopo l'eq. 7 in [12, sez. 3], viene posto $m = 2$ senza specificare il motivo di tale scelta.

4.1 M/M/1 vs m×M/M/1

Il modello che più si addice alla descrizione del processo di mining è m×M/M/1, ovvero un modello in cui ci sono m minatori, i quali hanno ciascuno una coda M/M/1. Se supponiamo che ogni coda abbia un tasso degli arrivi pari a λ e un tasso di servizio pari a μ, allora il modello complessivo è ancora markoviano e ha tassi rispettivamente pari a $\frac{\lambda}{m}$ e $\frac{\mu}{m}$.

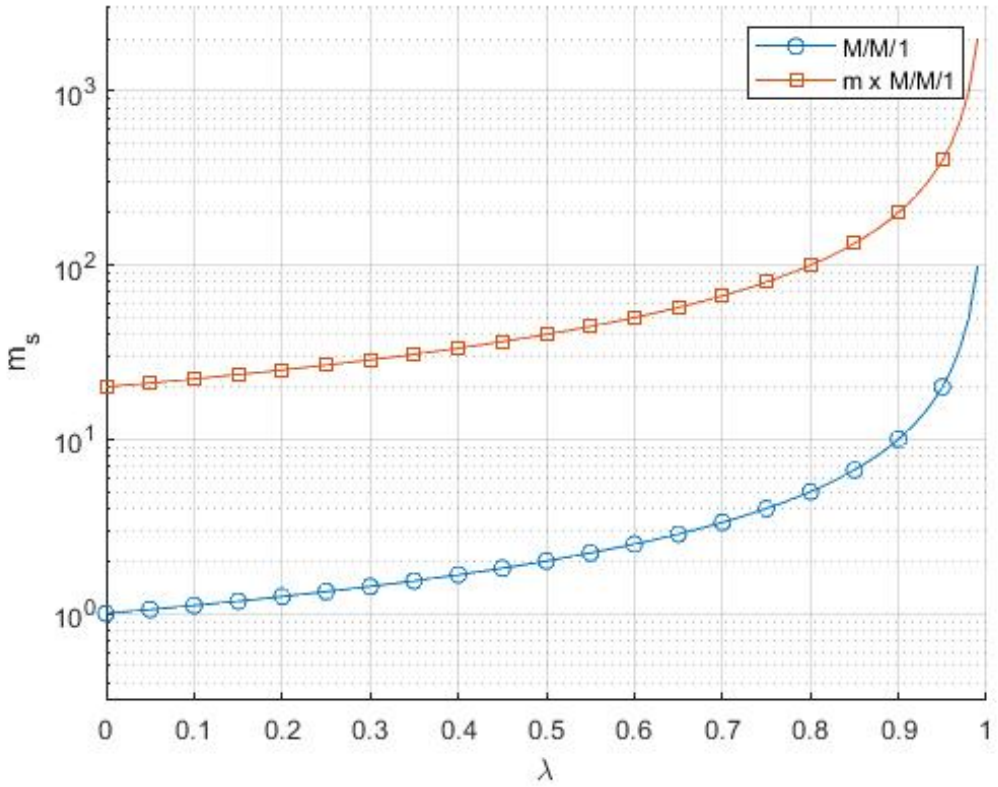

Figura 4.1: Tempo medio di sistema m_s in funzione del tasso degli arrivi λ dei due modelli, assumendo $m = 20$ e $\mu = 1$

Per osservare le prestazioni dei due modelli, è utile confrontare i loro tempi medi totali m_s, che sono pari a [2, p. 564]:

- per M/M/1

$$m_{s_1} = \frac{1}{\mu - \lambda} \tag{4.1}$$

- per m×M/M/1

$$m_{s_2} = \frac{m}{\mu - \lambda} \tag{4.2}$$

Nella fig. 4.1 è mostrato il grafico che confronta le prestazioni dei due modelli, assumendo $m = 20$ e $\mu = 1$. Il grafico mostra l'andamento del tempo medio totale m_s in funzione del tasso degli arrivi λ. Si può notare che m×M/M/1 è la soluzione peggiore in quanto, avendo m minatori che lavorano in parallelo, è altamente probabile che qualcuno di loro sia impegnato e qualcun altro non lo sia. Nel caso M/M/1, invece, la coda è unica e quindi

l'unico minatore che c'è, rimane sempre impegnato finchè è presente qualcuno in coda. Dunque, una transazione (o un blocco) richiede un tempo complessivo maggiore con il modello m×M/M/1 perchè, oltre ad aspettare in coda, deve attendere un tempo medio più lungo in servizio perchè, per ipotesi, il minatore è lento (ha un tasso di servizio pari a $\frac{\mu}{m}$ anzichè μ).

4.2 Analisi del processo delle partenze

Un altro vantaggio dell'utilizzo del modello M/M/1, risiede in un teorema che si riferisce al processo delle partenze da un sistema M/M/1 stabile. Questo teorema è il seguente.

Teorema 1 (di Burke)**.** *Il processo delle partenze da un sistema M/M/1 stabile è poissoniano di parametro* λ.

Il fatto che il tasso di uscita sia λ è dovuto alla stabilità del sistema. Il teorema di Burke dice una cosa più forte, ovvero che il processo delle uscite è poissoniano, quindi senza memoria. Questa proprietà può essere utilizzata per descrivere, sempre con un modello M/M/1, il *consenso*, ovvero quel processo che avviene dopo il mining. Quindi, possiamo rappresentare i due processi di mining e consenso come una *cascata di due code M/M/1 stabili e indipendenti*, in cui valgono tutte le proprietà già note sui singoli sistemi. Questo, è sicuramente un beneficio rispetto all'utilizzo di altri modelli, nel caso in cui si voglia analizzare (indipendentemente dal mining) anche il consenso.

5. Conclusioni

In questa tesi, si sono volute analizzare le statistiche dei tempi di conferma delle transazioni e gli aspetti che le influenzano. Inoltre, si sono confrontati i risultati degli articoli presenti in letteratura per avere delle soluzioni più ampie. In particolare, si è visto che:

- per M/G/1, l'analisi risulta più completa considerando un *batch service* anziche un servizio normale. Per questo modello, si sono studiati i fattori che impattano nei tempi di conferma e nei ritardi delle transazioni, concludendo che transazioni con alte commissioni o elevati importi, hanno dei tempi minori. Nel caso particolare di M/D/1, la durata del tempo di servizio è un compromesso dovuto alle possibili vulnerabilità che scaturiscono nel caso di tempi troppo lunghi o troppo corti;
- per M/M/1, c'è il vantaggio di poter studiare con lo stesso modello anche il processo del consenso; inoltre, in termini di prestazioni temporali, è preferibile un'unica coda M/M/1 con tasso di servizio μ anzichè m code con tasso $\frac{\mu}{m}$ ciascuna.

Lo studio delle criptovalute è tutt'ora un tema aperto, che può portare molti spunti interessanti per il futuro, soprattutto se un giorno, si decidesse di passare dai contanti alle criptomonete. La quasi totalità degli articoli presenti in letteratura, approfondisce soprattutto il processo di mining, che indubbiamente è il processo alla base delle monete digitali. Per studi futuri, si può pensare di ragionare anche sul processo del consenso, integrando lo studio con altri modelli di code.

Bibliografia

[1] Andreas M. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. 1st. O'Reilly Media, Inc., 2014. ISBN: 9781449374044 (cit. alle pp. 1, 4, 9).

[2] Nevio Benvenuto e Michele Zorzi. "Principles of Communications Networks and Systems". In: *Principles of Communications Networks and Systems* (set. 2011). DOI: 10.1002/9781119978589 (cit. alle pp. 3, 15, 18, 19, 24, 25, 27, 28).

[3] Bitcoin.org. *Introduction to Bitcoin Blockchain*. 2014. URL: https://bitcoin.org/en/blockchain-guide (cit. alle pp. 10, 14).

[4] data.bitcoinity.org. *Average time to mine a block in minutes*. 2019. URL: https://data.bitcoinity.org/bitcoin/block_time/all?f=m10&t=l.

[5] S. Ghimire e D. H. Selvaraj. "A Survey on Bitcoin Cryptocurrency and its Mining". In: *2018 26th International Conference on Systems Engineering (ICSEng)*. Dic. 2018, pp. 1–6. DOI: 10.1109/ICSENG.2018.8638208 (cit. alle pp. 1, 3, 6, 10–14).

[6] Mor Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139226424 (cit. a p. 19).

[7] Shoji Kasahara e Jun Kawahara. "Priority Mechanism of Bitcoin and Its Effect on Transaction-Confirmation Process". In: (mar. 2016) (cit. alle pp. 1, 22–25).

[8] Quan-Lin Li, Jing-Yu Ma e Yan-Xia Chang. "Blockchain Queueing Theory". In: *CoRR* abs/1808.01795 (2018). arXiv: 1808.01795. URL: http://arxiv.org/abs/1808.01795 (cit. alle pp. 1, 27).

[9] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *Cryptography Mailing list at https://metzdowd.com* (mar. 2009) (cit. alle pp. 1, 5, 9, 13).

[10] R. Raju, M. SaiVignesh e K. I. A. Prasad. "A Study of Current Cryptocurrency Systems". In: *2018 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*. Mar. 2018, pp. 203–209. DOI: 10.1109/ICCPEIC.2018.8525166 (cit. alle pp. 1, 11, 14).

[11] Saulo Ricci et al. "Learning Blockchain Delays: A Queueing Theory Approach". In: *SIGMETRICS Perform. Eval. Rev.* 46.3 (gen. 2019), pp. 122–125. ISSN: 0163-5999. DOI: 10.1145/3308897.3308952. URL: http://doi.acm.org/10.1145/3308897.3308952 (cit. alle pp. 1, 20–23).

[12] Riktesh Srivastava. "Blockchain and transaction Processing time Using M/M/1 Queue Model". In: *International Journal of Recent Technology and Engineering* (2019) (cit. alle pp. 1, 27).

[13] Alberto Zen. "Bitcoin - Analisi Tecnica ed Economica". Tesi di Laurea. Università Ca' Foscari Venezia, 2015 (cit. alle pp. 4, 10, 14).