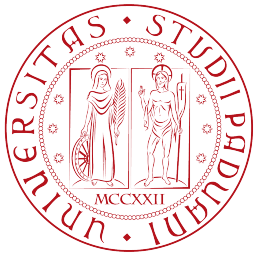

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria Informatica

Implementazione dell'algoritmo eMana per la sicurezza della rete IOTA

Relatore:
Prof. Alessandro Brighente
Correlatore:
Prof. Mauro Conti

Laureando:
Enrico D'Alberton
1225225

Anno Accademico 2021/2022

Abstract

La rete IOTA attualmente implementa un sistema di reputazione dei suoi nodi attraverso l'utilizzo di un token denominato "Mana". Di recente, in letteratura è stato teorizzato un nuovo sistema di reputazione chiamato "eMana" che tiene in considerazione molteplici aspetti e comportamenti di un nodo nella rete. In particolare, eMana si basa su sei sotto-algoritmi, i quali valutano non solo il numero di token IOTA movimentati e posseduti da un nodo, ma anche altre caratteristiche comportamentali dello stesso. In questa tesi, proponiamo di testare e validare eMana tramite una valutazione sperimentale. Adattiamo, quando possibile, le funzioni proposte al corrente sistema di valutazione che si suddivide in "consensusMana" e "accessMana". Per l'esperimento, è stato utilizzato GoShimmer, un software realizzato in linguaggio GoLang da IOTA Foundation stessa, che permette di eseguire una rete privata di test tramite un tool specifico. Quest'ultima, essendo una testnet, supporta delle funzionalità sperimentali leggermente diverse dalla rete principale, ma in questo caso è comunque un valido strumento di testing. A seguito della raccolta dei dati, i risultati ottenuti suggeriscono una conferma di quelle che erano le aspettative iniziali ipotizzate durante l'ideazione dell'algoritmo eMana. In particolare, questo nuovo sistema di reputazione propone una soluzione al problema del monopolio del Mana, a possibili attacchi Eclipse e allo spamming nella rete.

Indice

Capitolo 1

IOTA

In questo capitolo presentiamo la rete IOTA. Si affrontano nel dettaglio le dinamiche principali attraverso le quali i nodi interagiscono tra di loro, i dettagli del network e le sue funzionalità, al fine di comprendere a pieno l'esecuzione dell'esperimento.

1.1 Network

IOTA è un registro distribuito (in inglese comunemente detto *distributed ledger*) open-source realizzato con lo scopo di soddisfare le esigenze di network popolati da dispositivi IoT [1] che vengono identificati come *nodi*. Con il termine IoT si intendono dispositivi facenti parte del cosiddetto *Internet of Things* e non sono altro che oggetti fisici dotati di connessione in grado di comunicare a gruppi tra di loro. La nascita di IOTA risale al 2015, ma il primo white paper [2] messo in circolazione da Serguei Popov, uno dei fondatori, risale al 3 Aprile 2016. IOTA è anche il nome della criptovaluta attraverso la quale viene trasferito valore nella rete. Quest'ultima, a differenza degli altri protocolli, non utilizza una vera e propria blockchain per il salvataggio dei dati, ma implementa un sistema di storage delle informazioni differente, chiamato *Tangle*. Ciò permette alla rete di funzionare in modo totalmente decentralizzato, poiché ogni nodo che effettua una transazione deve convalidarne altre due precedentemente immesse nella rete. Questo permette al network di avere l'energia necessaria per rimanere in funzionamento e inoltre, si svincola dalla classica dualità tra utenti e miners presenti nelle classiche reti Proof of Work [3].

Nella rete il numero di token IOTA circolanti è pari a 2,779,530,283,277,761 e sono stati messi tutti in circolazione al momento della creazione del network. Que-

sto implica che non è possibile generare ulteriori token poiché il numero massimo è già stato raggiunto al momento della creazione.

1.2 Tangle

La Tangle è una struttura alternativa alla classica blockchain, difatti si basa su un grafo aciclico diretto, detto DAG, i cui vertici sono rappresentati dai messaggi, o transazioni, scambiati dagli utenti della rete. L'unica differenza tra messaggi e transazioni è che i primi non inviano valore, ma solo dati, mentre i secondi movimentano token IOTA. Ogni vertice del grafo fa riferimento a un vertice padre, quindi a un messaggio stanziato precedentemente. Inoltre, ogni collegamento tra i nodi del DAG è crittografato tramite la funzione di hash Ed25519 [4]. Nella figura 1.1 mostriamo una rappresentazione grafica della Tangle.

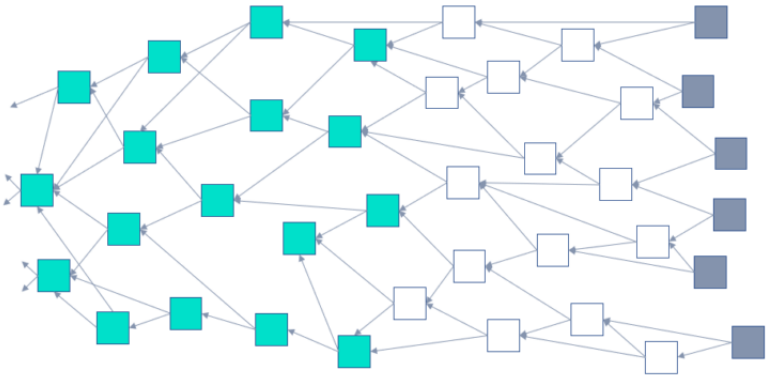

Figura 1.1: Tangle [5]

Una transazione viene riconosciuta valida dalla rete se soddisfa i seguenti punti:

- È sintatticamente corretta;
- Soddisfa le condizioni di UTXO [6];
- Non ci sono conflitti passati.

In questa tipologia di sistema distribuito, ogni nodo possiede una visione propria della rete, ossia una propria versione della Tangle. Infatti, ogni utente ne detiene una copia che viene costantemente aggiornata nel tempo grazie alla comunicazione con i nodi vicini. Di conseguenza, non esiste una visione univoca per l'intera rete e, ad esempio, una transazione comunicata a un capo del network potrebbe essere

recepita dagli altri nodi dopo diverso tempo. Lo stesso accade con i conflitti e in questo caso si parla di *branches*. Difatti, quando viene identificato un conflitto e bisogna prendere una decisione al fine di risolverlo, la rete si biforca creando dei branches, ossia più ramificazioni parallele nella Tangle. A questo punto gli utenti della rete risolvono il conflitto e l'ultimo branch rimasto sarà quello ritenuto valido e da cui continuerà la costruzione del grafo.

1.3 Wallet

Affinchè si possa interagire con il network IOTA è necessario possedere un *wallet*, che come suggerisce la parola stessa è assimilabile a un portafoglio digitale nel quale risiedono i nostri token. Per creare un wallet è sufficiente utilizzare un tool ufficiale creato da IOTA [7]. Il tool genera un *seed* casuale (una stringa alfanumerica lunga 81 caratteri) e da essa ricava le chiavi private da cui si generano gli indirizzi, detti *addresses*, e i node ID che sono più brevi e identificano il nodo nella rete. Ogni indirizzo è identificato da una stringa alfanumerica ed è utilizzato come riferimento per ricevere messaggi o transazioni (similmente a come si utilizzano gli IBAN nei bonifici bancari).

È di fondamentale importanza che l'utente conservi il proprio seed per sè, infatti la perdita di quest'ultimo comporta la perdita di tutti i fondi presenti nel relativo wallet e non c'è alcun modo per recuperarli. Citando un esempio concreto, nel network Bitcoin si stima che circa il 20% dei token esistenti si trovino all'interno di wallet di cui si ignora il seed e di conseguenza sono da ritenere persi per sempre [8].

1.4 Mana

Il network IOTA implementa un token secondario chiamato "Mana" che ha lo scopo di regolamentare l'utilizzo della rete e di evitare attacchi esterni che potrebbero comprometterne la stabilità. Il Mana si suddivide in due sottocategorie: *accessMana* e *consensusMana*. Il primo regola la congestione della rete, ossia il throughput del nodo, mentre il secondo regola la selezione dei nodi vicini e il meccanismo del consenso di transazioni e messaggi. Possedere più accessMana significa avere la possibilità di trasmettere più dati in meno tempo. Mentre, possedere più consensusMana significa avere più potere di voto riguardo l'approvazione di ciò che circola nella rete.

Tecnicamente un nodo può ottenere Mana in tre modi.

1. Detenendo dei token: i nodi possono acquistare token e delegare il mana a sé stessi.

2. Prendendo Mana da coloro che detengono dei token: i pagamenti possono essere effettuati in IOTA.

3. Processando il traffico di rete: un nodo può processare transazioni per ottenere il Mana derivante da essa.

Il quantitativo di Mana che viene delegato durante un trasferimento di token IOTA è calcolato in modo tale che un individuo malevolo non possa aumentare il Mana di un nodo a dismisura, evitando in questo modo il cosiddetto *Sybil Attack*.

1.5 Autopeering e selezione dei vicini

Affinchè vengano stabilite delle connessioni, un nodo ha bisogno di scoprire e trovare una lista degli indirizzi IP degli altri nodi. Quest'ultimi necessitano inoltre di dover mantenere aggiornato il proprio registro distribuito in modo tale che siano in grado di scambiare informazioni tra di loro (un registro datato potrebbe contenere dati non veritieri). Ogni nodo stabilisce un canale di comunicazione con un sottoinsieme (*neighbors*) di tutti i nodi della rete attraverso il processo di *peering*.

Tramite la *Peer Discovery* si riesce a ottenere una lista dei nodi verificati. Il processo consiste nel procurarsi una lista di nodi conosciuti e successivamente si validano attraverso il protocollo *Ping and Pong*. Quest'ultimo consiste nell'invio di un segnale *Ping* a un nodo da verificare e nella risposta con un segnale *Pong* da esso. Nel caso ciò non avvenga, non verrà inserito nella lista dei nodi verificati.

L'obiettivo della selezione dei vicini è di costruire un "vicinato" di nodi evitando di accettare possibili malintenzionati. I vicini sono accettati quando un nodo manda una *peering request* a un altro nodo, il quale può accettarla o rifiutarla con una *peering response*. Per prevenire gli attacchi, il protocollo crea una classifica dei possibili vicini in base al Consensus Mana posseduto e inoltre utilizza una funzione che casualmente mette in atto una verifica crittografica del nodo in questione. I nodi che terminano con successo questa selezione iniziano a far parte del vicinato. Con la figura 1.2 mostriamo un riassunto schematico di questo processo.

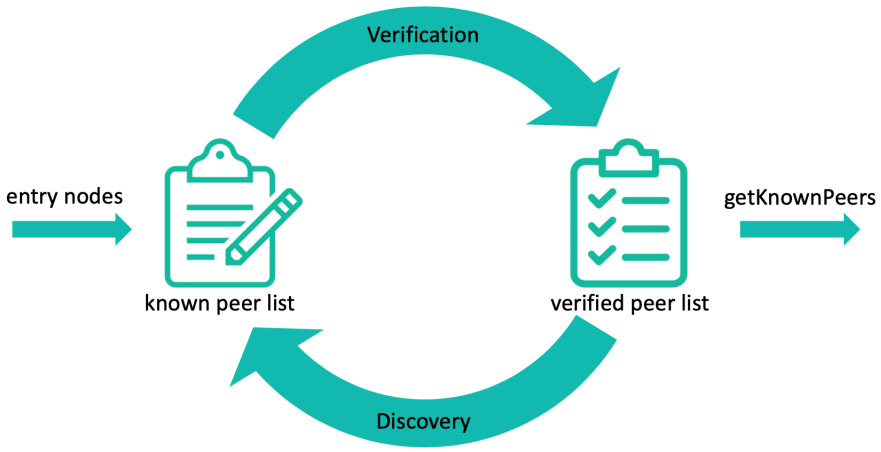

Figura 1.2: Peer discovery

Capitolo 2

Setup esperimento

In questo capitolo abbiamo lo scopo di descrivere gli strumenti utilizzati per l'esecuzione dei testi. In particolare, affronteremo il loro funzionamento ed esporremmo i dettagli delle reti nelle quali sono stati effettuati i test.

2.1 GoShimmer

GoShimmer è un progetto ingegneristico di ricerca sviluppato da IOTA Foundation con il fine di valutare nuovi concetti teorizzati, implementandoli in un software per l'esecuzione di un nodo. È quindi utilizzabile in una rete di test (*testnet*) che utilizza token privi di valore reale. L'obiettivo di IOTA Foundation con questo software è quindi quello di testare nuove funzionalità. Difatti, gli utenti per cui è stato pensato sono persone che possiedono delle conoscenze informatiche, piuttosto che individui "comuni". GoShimmer è realizzato in linguaggio Go [9], è eseguibile attraverso Docker [10] ed è in costante evoluzione, infatti il codice essendo open-source e contando 50 contributors su GitHub [11], viene aggiornato molto spesso. L'aspetto interessante di questo software sono i tool messi a disposizione dell'utente. In particolare il tool "Docker Private Network" permette di eseguire e configurare una rete di test locale IOTA con la quale è possibile interagire. Si può definire la quantità di token emessi nella rete, il numero di nodi e i loro seed, infine eseguendo uno script bash è possibile eseguire privatamente un network IOTA nella nostra rete domestica. Il software permette di ispezionare alcune informazioni attraverso delle porte che vengono mantenute aperte, infatti ogni nodo avrà un indirizzo nella rete e digitando su un qualsiasi browser il suo indirizzo seguito dal numero della porta (ad es. "172.178.0.11/8081") sarà possibile visualizzare dati e informazioni di quest'ultimo come MPS (Messages

Per Second), Bytes scambiati, memoria RAM utilizzata e molto altro. La porta 8081 fornisce una dashboard in cui sono presenti i dati del nodo interessato, inoltre è possibile avere una visione della Tangle che si espande in tempo reale e, accedendo alla porta 9000, una rappresentazione grafica dei nodi insieme ai loro vicini denominata *visualizer*. Nella figura 2.1 presentiamo la dashboard e, come si può notare, è visitabile alla porta 8081, mentre nella figura 2.2 riportiamo la rappresentazione grafica di alcuni nodi.

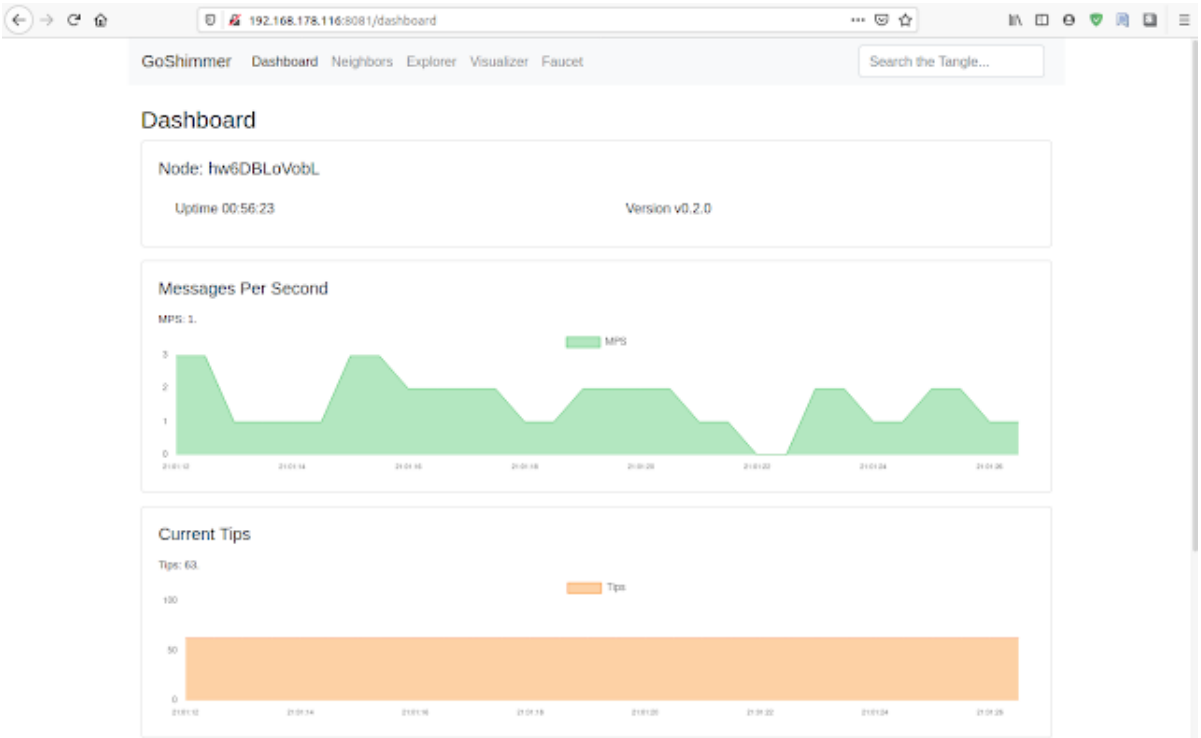

Figura 2.1: Dashboard disponibile alla porta 8081

Figura 2.2: Rappresentazione grafica dei nodi e delle loro connessioni (visualizer)

2.2 Implementazione Mana in GoShimmer

Come anticipato nel paragrafo precedente, in GoShimmer vengono testate nuove idee e funzionalità proposte da IOTA Foundation. Questo significa che nonostante i tools del software ci permettano di simulare una rete IOTA, le sue funzionalità saranno in parte differenti da quelle del network principale (*mainnet*). In particolare, trattando l'utilizzo del Mana in questa ricerca, risulta importante specificare l'implementazione di quest'ultimo all'interno di GoShimmer ed evidenziare le differenze con quella che è la mainnet.

Come già citato in precedenza, esistono due tipologie di Mana: accessMana e consensusMana. All'interno di GoShimmer sono stati implementati due ulteriori suddivisioni per il loro calcolo: *mana calculations 1* e *mana calculations 2*. Si parla di due metodi diversi per conteggiare il Mana dovuto ai nodi della rete e non dovrebbero essere implementati contemporaneamente nel network principale. Proprio per questo motivo sono stati implementati entrambi in GoShimmer, così da determinare quale performi meglio per poi utilizzarlo nella mainnet. Il primo metodo di calcolo assegna un numero di Mana pari a quelli movimentati in una transazione. Il secondo metodo di calcolo ha un'evoluzione prevedibile nel tempo, ossia non è influenzato da ulteriori transiti di token ed è creato da zero (a differenza del primo che veniva trasferito da un nodo all'altro).

In ogni transazione è dichiarato quanto accessMana e consensusMana cedere e a quali nodi. A partire da queste informazioni un nodo è in grado di calcolare localmente il *Base Mana Vector* da cui si ricava il *Base Mana* per entrambi i tipi di Mana appena citati. Un Base Mana Vector è composto da Base Mana 1

e Base Mana 2 e sono determinati attraverso i calcoli esposti in precedenza. Il Base Mana è trasferito da un nodo all'altro e questo lo rende di fatto una funzione discontinua nel tempo. Affinché possa essere resa continua, viene applicata una Media Mobile Esponenziale (detta EMA [12]) e da essa vengono ricavati i valori di Effective Base Mana 1 ed Effective Base Mana 2. La figura 2.3 rappresenta un riassunto grafico di quanto appena detto.

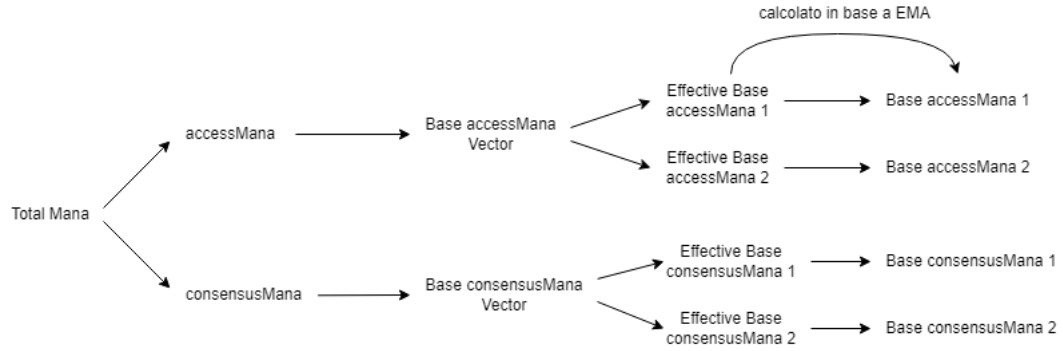

Figura 2.3: Schema calcolo del Mana in GoShimmer

2.3 Implementazione dell'algoritmo eMana nel codice

Tra le nuove idee provenienti dai ricercatori, è stato sviluppato un nuovo sistema di reputazione dei nodi che si svincolerebbe dalla già citata dualità del Mana. In questo paper [13], sviluppato da un gruppo di ricercatori dell'Università di Padova, è stato teorizzato un nuovo algoritmo in grado di regolamentare l'utilizzo del network IOTA attraverso l'implicazione di un solo token ausiliario che prende il nome di *eMana* (da *enhanced Mana*). Questo "giudica" i nodi non solo tramite i fondi movimentati, token IOTA, ma anche attraverso molteplici comportamenti riguardanti le relazioni con i vicini. L'algoritmo eMana è costituito da sei sotto-algoritmi:

- autopeeringvalidity;
- gossipparticipation;
- ratecontrolvalidity;
- congestioncontrolvalidity;

- fpcparticipation;
- msgvalidity.

La creazione di un nuovo token eMana all'interno di GoShimmer avrebbe rappresentato un cambio di struttura troppo significativo considerato che tutta la rete si basa sui calcoli del Mana esposti nella Sezione 2.2. Abbiamo puntato quindi a un adattamento dell'algoritmo eMana, senza che andasse a modificare la distribuzione del Mana, ma che si limitasse a giudicare i nodi secondo i criteri esposti nel paper. Abbiamo così costruito un sistema di reputazione dei nodi suddiviso in accessMana e consensusMana, si valuta cioè quanto un nodo sia degno di ricevere il Mana che normalmente gli verrebbe assegnato. Per realizzare tutto ciò, abbiamo suddiviso i sotto-algoritmi in base a quello che prendevano in considerazione nel nodo. Se si trattava di aspetti che normalmente venivano regolati dall'accessMana (ad es. controllo congestione della rete) allora avrebbero contribuito al calcolo di un livello di reputazione che avrebbe poi influenzato la distribuzione di accessMana. Al contrario, se la funzione valutava il nodo in base agli aspetti normalmente regolati dal consensusMana (ad es. selezione dei vicini), allora avrebbero contributo al calcolo di un diverso livello di reputazione. Questi due livelli non sono altro che numeri decimali compresi tra 0 e 1, inoltre se combinati insieme tramite una media ritornano la reputazione generale del nodo nella rete. Ad esempio, un nodo che ha tentato di occludere il network, verrà giudicato negativamente e avrà reputazione bassa per quanto riguarda l'accessMana, ma potrebbe essersi attenuto a un comportamento eccellente sotto altri aspetti e la sua reputazione relativa al consensusMana potrebbe essere molto alta, il risultato sarebbe una reputazione totale comunque discreta. Affinché ogni nodo sia indipendente dagli altri e seguendo il principio secondo cui nessuno può conoscere dati di altri nodi nella rete, abbiamo realizzato un sistema in cui ogni nodo calcola la reputazione dei propri vicini secondo il proprio punto di vista (come per la Tangle, ogni nodo possiede la propria). Questo significa che per calcolare un grado di reputazione a livello "globale" nella rete sarà necessario fare una media di tutti i gradi calcolati dai neighbors di quel nodo. Come già detto in precedenza, l'algoritmo implementato non fa variare la distribuzione del Mana, ma semplicemente valuta i nodi sulla base dei loro comportamenti in una rete privata simulata tramite il tool *Docker Private Network* di GoShimmer. Proponiamo di seguito una breve analisi e descrizione dei sei sotto-algoritmi esposti nel paper precedentemente citato:

1. autopeeringvalidity: blocca l'ingresso nel network ai nodi che hanno tentato di ingolfarlo. Contribuisce al livello di reputazione relativo al consensusMana.

2. gossipparticipation: contribuisce al livello di reputazione relativo al consensusMana ed è costituito da 3 sotto-algoritmi:

   (a) saltValidation: viene confrontato il salt dichiarato con il salt corrente. Questa funzione non è stata implementata a causa di limitazioni a livello software.

   (b) distancevalidity: valuta se due nodi sono troppo distanti.

   (c) selfish: valuta se un nodo è "egoista", ossia se tende a mantenere i messaggi per sé senza condividerli con i vicini.

3. ratecontrolvalidity: previene lo spam di messaggi nella rete. Contribuisce al livello di reputazione relativo all'accessMana. Questa funzione non è stata implementata a causa di limitazioni a livello software.

4. congestioncontrol: ha l'obiettivo di massimizzare il throughput e allo stesso tempo minimizzare i ritardi nella trasmissione. Contribuisce al livello di reputazione relativo al accessMana.

5. fpcparticipation: controlla che i nodi partecipino alle votazioni nel network (ad es. votazioni dei branch). Contribuisce al livello di reputazione relativo al consensusMana.

6. messagevalidity: contribuisce al livello di reputazione relativo al consensusMana ed è costituito da 3 sotto-algoritmi:

   (a) validatesign: controlla che la firma dei messaggi sia uguale all'hash della chiave pubblica del mittente.

   (b) validatemaxdepth: verifica che un messaggio non faccia riferimento a un messaggio padre troppo vecchio.

   (c) validatevaluemsg: controlla che i token in input siano stati stati totalmente "spesi" in output. Questa funzione non è stata implementata a causa di limitazioni a livello software.

Nella figura 2.4 proponiamo uno schema riassuntivo della classificazione degli algoritmi sopra citati.

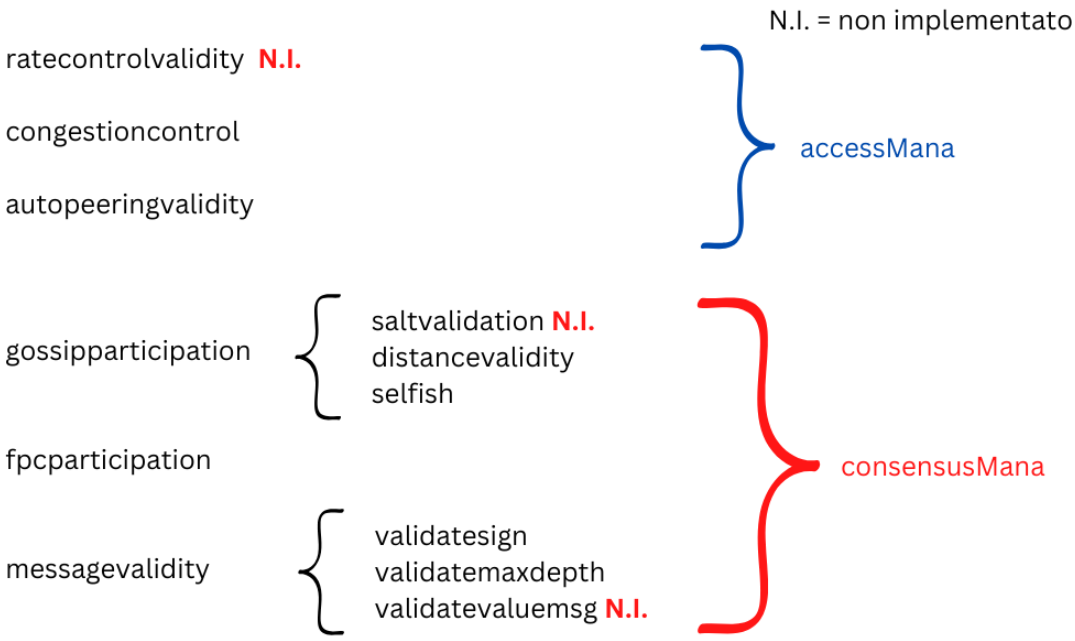

Figura 2.4: Schema algoritmi

2.4 Configurazione GoShimmer

Il funzionamento del Private Docker Network di GoShimmer viene gestito tramite diversi file di configurazione, in particolare è possibile decidere quanti token IOTA verranno inizialmente distribuiti, quanti Mana e a quali nodi. È inoltre possibile decidere l'ID dei nodi in modo che siano sempre gli stessi ogni volta che si genera la rete.

Per creare una rete con caratteristiche specifiche è necessario avvalersi di diversi tool offerti nel codice di GoShimmer. Innanzitutto, per eseguire i test, era importante che la rete disponesse di un numero definito di nodi che interagissero tra loro con gli stessi ID. Per realizzare ciò, abbiamo utilizzato un tool specifico chiamato *rand-seed* realizzato sempre in GoLang che è in grado di generare seed casuali, ma sintatticamente validi. In seguito, è stato necessario generare uno snapshot che potesse comunicare le informazioni di generazione della rete al tool docker-network. Per fare ciò abbiamo utilizzato il tool *genesis-snapshot*, nel quale abbiamo specificato il seed dei nodi da noi desiderati e il quantitativo di token da generare. Sostituendo lo snapshot ottenuto con quello precedente e modificando alcune impostazioni nel file *docker-compose.yml* relativo al tool docker-network, la rete era correttamente configurata.

Il risultato è un network nella quale sono presenti 7 nodi con i seguenti ID:

- 3oyiGRU8RTJ;
- HuhofQwrhYg;
- ZCYzMk7Wb5i;
- GYuQSRqLUt7;
- ZfkSwZ1kGgs;
- 4cmyArg99ug;
- 5BG6ZcE4QCX.

Nella rete viene distribuito tra i nodi un quantitativo di Mana pari a 100,000,000,000,000 con un'aggiunta di 1,000,000,000,000,000 al nodo master (4cmyArg99ug).

Capitolo 3

Risultati numerici

Nel seguente capitolo esponiamo i dati raccolti e proponiamo una loro analisi osservando come l'algoritmo sia in grado di rimediare ai problemi ipotizzati nel paper di riferimento, in modo da dimostrare che può avere una valenza e un'utilità per la rete IOTA.

3.1 Test eseguiti

In questo esperimento abbiamo condotto 12 test e altrettante misurazioni. Ogni test consisteva nel calcolo della reputazione di ogni nodo a livello globale nella rete. Per fare ciò abbiamo calcolato la media della reputazione di un nodo fornita dai suoi vicini relativa a ogni tipo di Mana. Successivamente abbiamo calcolato la media tra le due reputazioni appena ottenute. Ad esempio, nella figura 3.1 un nodo A possiede come vicini i nodi B, C e D. Ognuno di essi possiede una propria opinione sulla reputazione del nodo A suddivisa in aMana reputation e cMana reputation. Per ottenere la reputazione globale del nodo dobbiamo eseguire i calcoli esposti prima e mostrati in figura.

Inoltre, le funzioni *distancevalidity* e *selfish* hanno due threshold che possono essere fatte variare in modo che si adattino al meglio nella rete in cui agiscono. In questi test abbiamo provato tre valori di threshold per quando riguarda la funzione distancevalidity e due threshold per quanto riguarda la funzione selfish. Il numero di test così elevato è dovuto al fatto che abbiamo eseguito tutte le possibili combinazioni tra le threshold. I primi sei test hanno verificato il comportamento dei nodi prima, durante e a seguito di uno spam di transazioni durato 15 secondi con una frequenza di 50 MPS. Invece, gli ultimi sei test hanno verificato il comportamento dei nodi prima, durante e a seguito di uno spam di conflitti durato

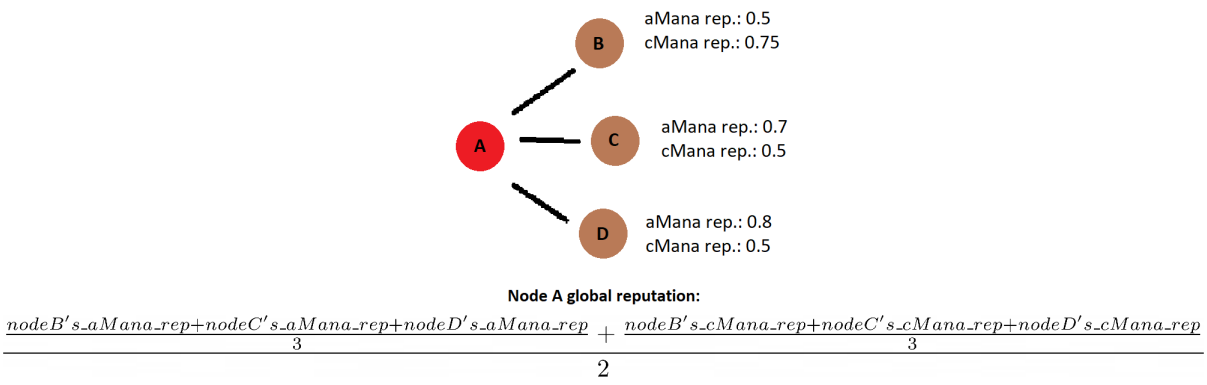

Figura 3.1: Schema algoritmi

15 secondi a una frequenza di 2 MPS. Complessivamente tra tutti i test abbiamo calcolato 273 reputazioni globali.

3.2 Analisi

Il paper di riferimento nel quale viene teorizzato il nuovo algoritmo di valutazione "eMana" mette alla luce alcune evidenti problematiche dell'attuale rete IOTA e si propone come possibile mezzo di risoluzione. Le criticità citate sono:

- Monopolio del Mana: nel sistema corrente il Mana viene calcolato in base al quantitativo di token IOTA movimentati attraverso le transazioni, di conseguenza nodi con più token avranno più diritto ad ottenere Mana.
- Attacchi Eclipse: l'attaccante mira a "soffocare" un nodo nella rete per poi immettere dati non veritieri nel network a sua insaputa.
- Associazioni basate sul Mana: nel processo di autopeering, nodi con lo stesso quantitativo di Mana tendono a interagire più facilmente. Inoltre, il fatto che attualmente il Mana venga calcolato in base alle movimentazioni dei propri fondi, non favorisce l'ingresso nella rete di nuovi nodi privi di token IOTA, i quali potrebbero comunque attenersi a un buon comportamento.
- Spamming: un grande quantitativo di richieste nella rete potrebbe portare al momentaneo disfunzionamento del network.
- Non completezza del sistema di valutazione: l'attuale sistema è in grado di giudicare i nodi solo su specifici comportamenti, non avendo così una visione complessiva dello stesso.

Per quanto concerne il monopolio del Mana e la sua non completezza come sistema di valutazione, è possibile affermare che una soluzione come quella di eMana può rappresentare un significativo cambiamento dell'intero network. Grazie ai dati ricavati dai test è possibile notare che l'algoritmo è sensibile a diversi aspetti comportamentali dei nodi, come ad esempio il cambiamento di frequenza dell'invio di messaggi o la partecipazione attiva nella rete. Questi aspetti, per quanto possano a prima vista sembrare banali, vanno a contribuire nella valutazione comportamentale di un nodo, creando così un sistema di valutazione più completo che eviti situazioni di monopolio del Mana, le quali potrebbero scoraggiare l'ingresso di nuovi nodi della rete.

Gli attacchi Eclipse vengono invece prevenuti anch'essi grazie all'algoritmo, in particolare a funzioni come *selfish* che sono in grado di valutare un nodo in base a quanto comunica con il resto del network. Difatti, effettuando dei test con diverse threshold relative alla funzione appena citata, i valori medi dei livelli di reputazione con una threshold più difficile da raggiungere sono stati inferiori del 3.62% rispetto a un'altra più permissiva nei confronti degli stessi. Questa percentuale è probabilmente destinata a crescere con l'aumentare del valore della threshold poiché i nodi avranno sempre più difficoltà a raggiungerla. Inoltre, azioni come il *double spending*, ossia il tentativo di effettuare pagamenti duplici con la stessa valuta, sono identificati dalla Tangle e creano dei conflitti nella rete. L'algoritmo, grazie alla funzione *fpcparticipation*, è in grado di identificare questi conflitti e giudica positivamente i nodi che prendono attivamente parte alla loro risoluzione. Questo comportamento è individuabile dai grafici 3.5, 3.6 e 3.7, infatti durante lo spam di conflitti, i nodi prendono parte alle votazioni per risolverli e la loro reputazione sale, per poi tornare a un valore più basso di quello iniziale, probabilmente dovuto all'elevato numero di messaggi che viene momentaneamente identificato dall'algoritmo come spam e quindi penalizzato.

Gli attacchi di spamming sono probabilmente i più comuni e permettono di bloccare l'intero network anche per diverse ore. Considerando che la rete IOTA è stata studiata per i dispositivi IoT, la capacità di difendersi da questo tipo di attacco risulta sicuramente importante (basti pensare a future implementazioni di questi dispositivi in ambito medico). Nei test di spamming delle transazioni eseguiti tramite GoShimmer è stata evidenziata una diminuzione della reputazione dei nodi durante lo spam, ciò è osservabile nei grafici 3.2, 3.3 e 3.4. Questo va a testimonianza del fatto che il nuovo sistema di reputazione è in grado di identificare eventuali anomalie nella rete e di limitare i comportamenti dei nodi

nel network a seguito di ciò.

I grafici esposti sono relativi ai test eseguiti tramite GoShimmer e descritti nel paragrafo 3.1.

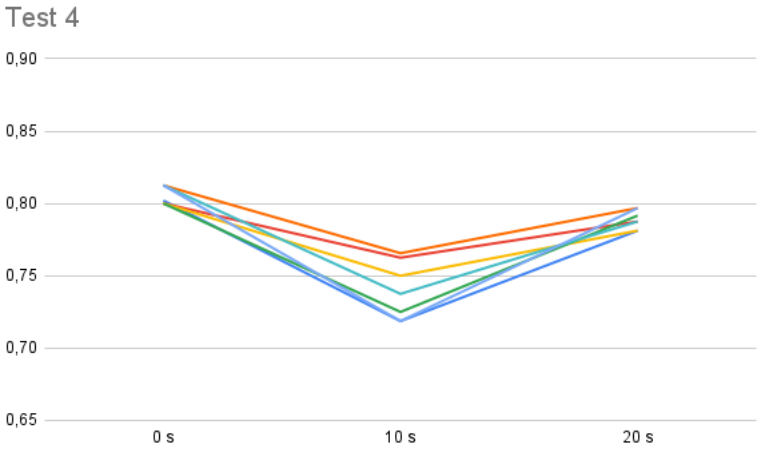

Figura 3.2: Grafico reputazione durante spam di transazioni (50MPS, 15s), distanceValidity threshold=2158898094, selfish threshold=2

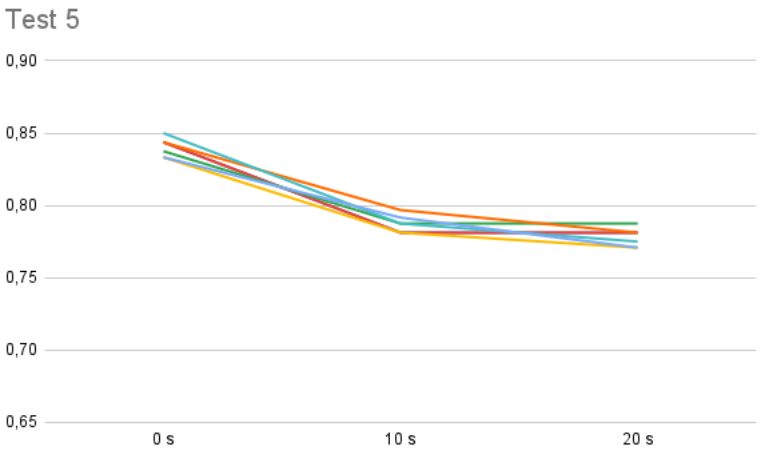

Figura 3.3: Grafico reputazione durante spam di transazioni (50MPS, 15s), distanceValidity threshold=2452019764, selfish threshold=2

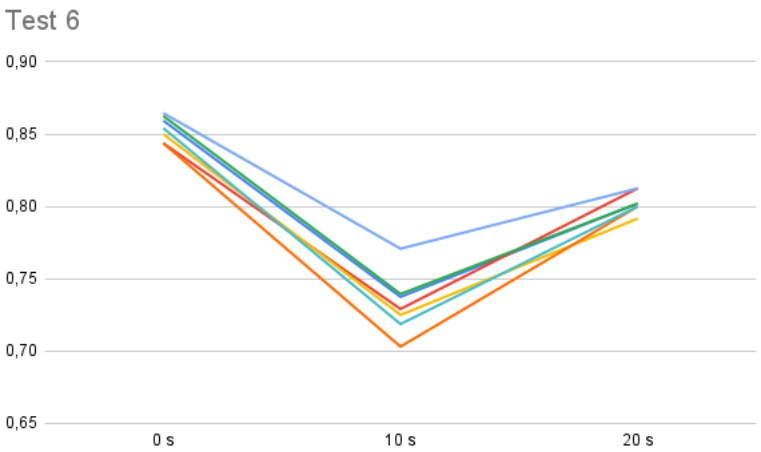

Figura 3.4: Grafico reputazione durante spam di transazioni (50MPS, 15s), distanceValidity threshold=2782705544, selfish threshold=2

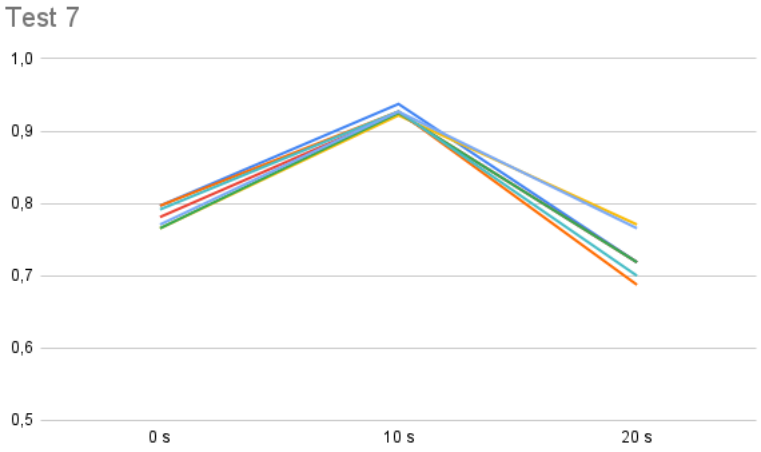

Figura 3.5: Grafico reputazione durante spam di transazioni (50MPS, 15s), distanceValidity threshold=2452019764, selfish threshold=1

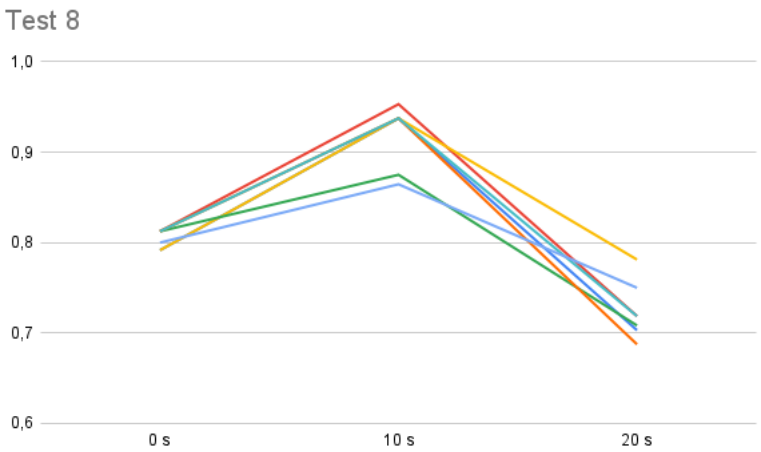

Figura 3.6: Grafico reputazione durante spam di transazioni (50MPS, 15s), distanceValidity threshold=2452019764, selfish threshold=1

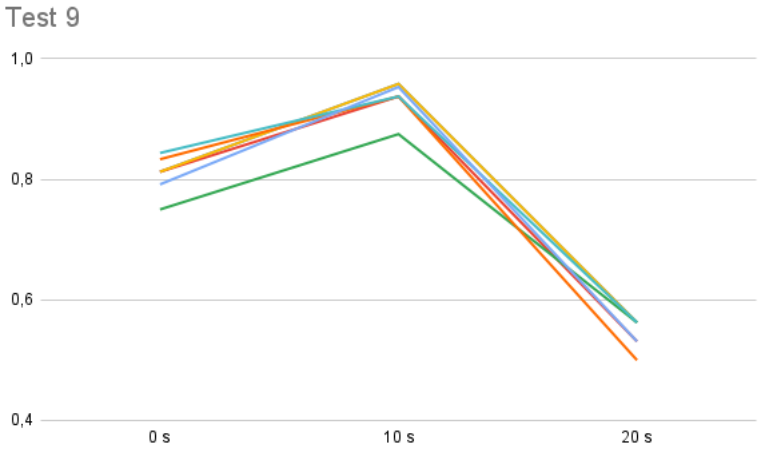

Figura 3.7: Grafico reputazione durante spam di transazioni (50MPS, 15s), distanceValidity threshold=2782705544, selfish threshold=1

Capitolo 4

Conclusione

Nel seguente capitolo traiamo le conclusioni a seguito dei risultati e delle analisi esposte nel Capitolo 3.

4.1 Conclusione

Le possibili implicazioni dei risultati ottenuti dai test sono diverse. Questi esperimenti avevano l'obiettivo di verificare la validità delle ipotesi avanzate dal paper di riferimento in cui si presentava eMana. Di conseguenza, lo scopo finale è rimasto quello di validare un sistema di valutazione che si basi su quello precedente migliorandone alcuni aspetti. Dai risultati evidenziati risulta chiaro che un sistema come eMana potrebbe facilitare l'ingresso di nuovi nodi nella rete, incoraggiarli a determinati comportamenti che aumenterebbero la sicurezza del network stesso e al contempo proteggerla da spamming e attacchi come il double spending.

Bibliografia

[1] Rose, K., Eldridge, S. & Chapin, L. The internet of things: An overview. *The Internet Society (ISOC)*. **80** pp. 1-50 (2015)

[2] Popov, S. The tangle. *White Paper*. **1** (2018)

[3] Nakamoto, S. Bitcoin whitepaper. *URL: Https://bitcoin. Org/bitcoin. Pdf-(: 17.07. 2019)*. (2008)

[4] Bernstein, D., Duif, N., Lange, T., Schwabe, P. & Yang, B. High-speed high-security signatures. *Journal Of Cryptographic Engineering*. **2**, 77-89 (2012)

[5] *TANGLE — IOTA BEGINNERS GUIDE*, `https://iota-beginners-guide.com/dlt/tangle/` **2022**

[6] *Wiki IOTA - UTXO*, `https://wiki.iota.org/IOTA-2.0-Research-Specifications/5.1UTXO`

[7] *Official IOTA Wallet*, `https://github.com/iotaledger/firefly`

[8] *Tens of billions worth of Bitcoin have been locked by people who forgot their key*, `shorturl.at/bdh08` **Jan., 2021**

[9] *The Go Programming Language*, `https://go.dev/`

[10] *Docker*, `https://www.docker.com/`

[11] *GoShimmer - GitHub page*, `https://github.com/iotaledger/goshimmer`

[12] Wikipedia contributors Exponential Moving Average — Wikipedia, The Free Encyclopedia. (2007), https://en.wikipedia.org/w/index.php?title=Exponential$_{M}oving_{A}verageoldid = 105891727, [*Online*; *accessed*4 − *September* − 2022]

[13] Saha, R., Kumar, G., Brighente, A. & Conti, M. Towards An Enhanced Reputation System for IOTA's Coordicide. *2021 Third International Conference On Blockchain Computing And Applications (BCCA)*. pp. 26-33 (2021)

[14] Helmer, L. & Penzkofer, A. Report on the energy consumption of the IOTA 2.0 prototype network (GoShimmer 0.8. 3) under different testing scenarios.