POLITECNICO DI TORINO

Collegio di Ingegneria Meccanica, Aerospaziale,

dell'Autoveicolo e della Produzione

Corso di Laurea Magistrale in Ingegneria

della Produzione Industriale e dell'Innovazione Tecnologica

Tesi di Laurea Magistrale

IoT e DLT per il monitoraggio di una supply chain: un'app di tracking con IOTA

Candidato:	**Relatori**:	**Tutor Aziendali:**
Federico Facelli	Guido Perboli	Umberto Sgueglia
	Stefano Musso	Domenico Scalera

Anno accademico 2019-2020

Ringraziamenti

Alla fine di questo percorso durato 18 anni, che segna anche la fine di un periodo importante della mia vita, ci tengo a ringraziare per primi i miei genitori, che mi hanno sempre appoggiato nelle scelte che ho fatto spronandomi a dare il massimo, da quando suonavo in discoteca e mi venivano a prendere fino ai due anni passati all'estero. Ringrazio inoltre mio fratellino, compagno di risate, di scherzi e vicino a me da 20 anni, nonostante durante gli ultimi 4 abbiamo spesso vissuto distanti. Ringrazio Maria, per le avventure e l'amore che mi ha dato da quando l'ho conosciuta e per i voli che ha preso spesso per venirmi a trovare a Barcellona (non era poi così male). Grazie a tutta la mia famiglia: zio Mau e Fede, zia Betti e Cri, i nonni Lella, Toio e nonna Ziu (rimettiti presto) e la mia cuginetta Bea, tutti fonte costante d'ispirazione per me e di ricordi indelebili e molto belli. Ringrazio tutti i miei amici: il gruppo III, gli amici di sempre, punto fermo sui quali posso contare per qualsiasi cosa, partner di avventure e risate; i KS che in questi anni di studio e divertimento sono stati i migliori compagni che potessi immaginare; Olek e Guggi, amici di progetti (dalla radio agli eventi e oltre) coi quali ho svarionato e conosciuto Alex Catania (Guggi grazie per i consigli su Iota). Grazie a Blue Reply nelle persone di Umberto Sgueglia, Domenico Scalera, Andrea Venditti e Vincenzo Stola per l'opportunità e l'enorme supporto che mi hanno fornito in questi mesi di stage. Un grazie infine a Guido Perboli e Stefano Musso per la disponibilità e l'aiuto fornitomi in qualità di relatori.

INDICE

1. Introduzione 1
1.1. Supply Chain: struttura 5
1.2. Supply Chain: criticità e digitalizzazione 8
1.3. DLT e Blockchain: funzionamento base 11
1.4. IoT e DLT in una Supply Chain: benefici 16
1.5. Step e requisiti per l'implementazione 19
1.5.1. Requisiti 19
1.5.2. Step per l'implementazione 22
2. DLT e Blockchain: tecnologia, funzionamento e potenzialità 24
2.1. Origini: Bitcoin 25
2.2. Modalità 26
2.2.1. Blockchain Pubbliche 27
2.2.2. Blockchain Private 27
2.3. Funzionamento base 28
2.3.1. La scalabilità della Rete e i Fork 30
2.3.2. Possibili minacce alla rete 31
2.3.3. Algoritmi di Consenso 35
2.4. Funzionalità 42
2.4.1. Tokenizzazione 42
2.4.2. Smart Contracts 44
2.5. Casi d'uso 48
3. Valutazione idoneità del framework DLT 51
4. IOTA: DLT per l' Internet of Things 56
4.1. Componenti del Tangle 58
4.1.1. La rete delle Transazioni di IOTA 58
4.1.2. La rete dei Nodi di IOTA 60
4.1.3. Velocità del Consenso 62
4.1.4. Sicurezza 63
4.1.5. Snapshot 64

4.1.6. Wallet 64
4.2. Funzionamento delle transazioni nel tangle 69
4.3. Masked Authenticated Messaging (MAM) 80
4.3.1. Canali MAM 80
4.3.2. Merkle Tree 81
4.3.3. Catena di messaggi 84
4.3.4. Struttura di un pacchetto MAM 85
4.3.5. Indirizzi 86
4.3.6. Sequenza di pubblicazione 86
4.3.7. Proprietà del canale 86
4.3.8. Sezione MAM 87
4.3.9. Sezione della firma (Signature Section) 88
4.3.10. Recuperare i dati (MAM Fetch) 89
4.4. Sviluppi Futuri 90
4.4.1. Coordicide 90
4.4.2. Smart Contracts 91
4.4.3. Streams 93
5. PoC: "track and trace" in una Supply Chain con IOTA 95
5.1. Caso d'Uso 95
5.2. Struttura della Rete 97
5.3. Sviluppo della Demo 104
5.3.1. MAM API 104
5.3.2. Publish e Fetch 107
5.4. Analisi delle Prestazioni 112
6. Conclusioni 120
Annex **126**
Indice delle Figure **146**
Indice dei Grafici **148**
Indice delle Tabelle **149**
Bibliografia **150**

1. Introduzione

Di fronte alla crisi sanitaria senza precedenti causata dalla pandemia di Covid-19, l'attenzione delle aziende di tutto il mondo si è giustamente concentrata prima di tutto sulla sicurezza dei dipendenti, costringendo alcuni stabilimenti e uffici alla chiusura. I produttori delle industrie critiche rimaste in attività sono stati costretti a trovare fornitori alternativi per mantenere in funzione le fabbriche, viste le limitazioni doganali imposte. Negli ultimi mesi, mentre in alcune regioni la fase più acuta della pandemia sembra superata e le abitudini dei cittadini si sono adattate alle nuove normative, nelle imprese è iniziata la pianificazione della ripresa a partire da valutazioni sull'affidabilità e il rischio rappresentano le supply chain.

Inoltre, è importante non considerare la pandemia in corso come un evento isolato. Le interruzioni stanno aumentando in frequenza e ampiezza per cause di diverso genere: eventi geopolitici, disastri legati al clima e crisi di salute pubblica sono una costante negli ultimi anni (la Brexit e la guerra commerciale USA-Cina sono alcuni esempi). Per decenni, i principi cardine alla base del corretto funzionamento e della massima efficienza nelle supply chain di tutto il mondo sono stati **l'approvvigionamento a basso costo** e **l'inventario minimo**. Ma in un sistema internazionale sempre più turbolento, le reti eccessivamente dipendenti da queste variabili rischiano di diventare troppo vulnerabili e poco flessibili.

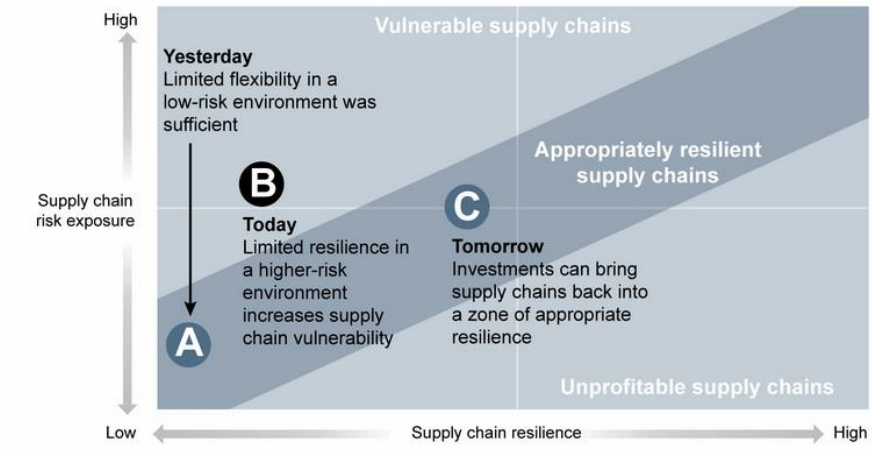


Grafico 1-rapporto tra esposizione al rischio e resilienza nelle supply chain di ieri, oggi e domani

Reti più flessibili aiutano le aziende ad adattarsi rapidamente nei periodi di alta variabilità de intensità della domanda, permettendo ai team di produzione di muoversi agilmente e adattarsi, apportando un vantaggio competitivo significativo.

Un' analisi di Bain mostra come le aziende dotate di supply chain resilienti crescono più velocemente perché in grado di muoversi rapidamente per soddisfare e adattarsi alle esigenze dei clienti. Questo tipo di flessibilità nella struttura permette di tagliare i costi e migliorare il flusso di cassa, grazie ad un aumento del 10-40% dell'indice di rotazione delle merci. Infine, le interruzioni vengono ridotte costruendo buffer lungo tutta la rete di fornitura e investendo in analisi avanzate per migliorare la pianificazione e la precisione delle previsioni.

Le imprese che hanno subito un'improvvisa carenza di materie prime o parti di ricambio sanno quanto possa essere costoso sottovalutare quest'aspetto al giorno d'oggi. Un produttore globale di elettronica ha subito un calo del 16% del fatturato e un calo del 66% dell'utile netto nel 2016, quando una serie di terremoti a Kumamoto, in Giappone, ha interrotto la fornitura di parti. E per una casa automobilistica globale dipendente da parti prodotte in Tailandia, le inondazioni del 2011 hanno causato un calo del 5% della produzione globale, equivalente a 5 miliardi di dollari di vendite perse.

In compenso, molte aziende stanno investendo nella resilienza della supply chain per ridurre al minimo tali rischi e per aumentare l'efficienza. Prendiamo l'esempio di Toyota che ha investito in nuovi strumenti per migliorare la resilienza della supply chain. Lavorando con i suoi partner, l'azienda ha creato un database per visualizzare le reti di fornitura per ogni componente. In caso di disastro, Toyota è in grado di identificare immediatamente i componenti a rischio. Il database identifica le componenti che sono fornite da un solo produttore e che sono difficili da sostituire. In questo modo l'azienda è in grado di ridurre drasticamente la dipendenza da singoli fornitori.

Le aziende che investono nella resilienza della propria supply chain riducono i cicli di sviluppo dei prodotti del 40-60% e sono in grado di espandere la capacità produttiva del 15-25% ottimizzando le operazioni.

Questo fatto è stato ulteriormente sottolineato dalla crisi del Covid-19, che ha puntato i riflettori sulle aziende che hanno già linee di produzione flessibili. Infatti, quando la pandemia ha travolto il sistema sanitario francese e italiano e le forniture mediche hanno iniziato a scarseggiare, i produttori di beni di lusso agili hanno prontamente revisionato le operazioni interne per realizzare gli articoli di cui c'era urgente bisogno. Le supply chain flessibili hanno svolto un ruolo fondamentale, permettendo l'approvvigionamento rapido delle materie prime, la progettazione, lo sviluppo e il collaudo dei prodotti e la loro distribuzione.

Considerando ciò che abbiamo visto, si possono identificare cinque principali capacità, necessarie al fine di creare una supply chain resiliente al variare delle condizioni che la circondano:

- *Agilità della rete*: per reagire rapidamente alle interruzioni è fondamentale la presenza di un ecosistema flessibile di fornitori e partner in grado di gestire carenze improvvise o addirittura di realizzare nuovi prodotti, adattando la struttura della linea. Ciò significa disporre di siti di produzione e nodi di assemblaggio alternativi e sfruttare al meglio gli strumenti messi a disposizioni dalle innovazioni dall'Industria 4.0 per ottimizzare i costi, migliorare la visibilità sulla rete e accelerare i tempi di reazione. Per

renderne possibile l'attuazione, è dunque necessario che all'interno delle supply chain le aziende leader sviluppino soluzioni su misura per ogni segmento.

- *Collaborazione digitale*: le applicazioni basate sul cloud e le piattaforme e di collaborazione contribuiscono a facilitare la condivisione delle informazioni in un ambiente sicuro, oltre a migliorare la qualità e la velocità del processo decisionale all'interno di un'organizzazione, con i fornitori e altri partner.

- *Visibilità della rete in tempo reale*: la tecnologia 5G, la *blockchain* e i dispositivi dell'Internet of Things (IoT) sono esempi di soluzioni che, in particolar modo se utilizzate contemporaneamente, rendono possibile una visibilità dell'intera supply chain in tempo reale. Le aziende sono in questa maniera in grado di calibrare l'offerta in base alla domanda prevista confrontando i dati sulla capacità di produzione interna con i segnali che arrivano in tempo reale.

- *Una rapida generazione di insight*, per rimanere un passo avanti rispetto alle interruzioni e migliorare la capacità di analizzare rapidamente i dati interni e le fonti esterne di grandi dati. Ciò può essere possibile sfruttando l'apprendimento automatico e l'intelligenza artificiale per l'analisi predittiva e prescrittiva.

- *Team decentralizzati* che possano reagire prontamente alle informazioni fornite, aiutando le aziende a uscire indenni dai periodi di interruzione e dalle oscillazioni.

All'interno della presente tesi, svolta all'interno della business unit IoT di Blue Reply srl, è stata esplorata l'applicazione di IoT e Distributed Ledger Technologies (DLT), per testarne il funzionamento e per comprendere sotto quali aspetti questi contribuiscano a rendere una supply chain più resiliente e adattabile a variazioni inattese. Nel fare ciò, sono state esaminate a livello generale le componenti principali delle supply chain odierne, per poi spostare l'attenzione sul funzionamento delle DLT (facendo riferimento in particolare ad alcuni framework di blockchain) e sulla loro applicazione al contesto; successivamente, è stata stilata una lista di condizioni che devono sussistere affinché possa considerarsi necessario l'utilizzo di queste tecnologie e step implementativi che devono essere rispettati nell'introduzione della soluzione in un caso d'uso reale, fornendo indicazioni su quali siano i parametri da tenere in considerazione per selezionare il framework che si addice maggiormente ad esso. Infine, dopo aver spiegato nel dettaglio il funzionamento della tecnologia DLT scelta, sono state illustrate architettura e funzionalità della PoC sviluppata, sottolineandone le peculiarità e analizzandone le prestazioni, traendo da qui le conclusioni.

1.1. Supply Chain: struttura

Con un'economia sempre più volta a soddisfare i bisogni del cliente, è ormai indispensabile monitorare, anticipare e controllare i l'intera vita dei prodotti, dalla produzione e dal trasporto fino e oltre alla vendita, prevedendo l'evoluzione dei consumi e delle tendenze del mercato.

Tutti gli individui, le organizzazioni, le risorse, le attività e le tecnologie coinvolte nella creazione e nella vendita di un prodotto sono parte della struttura di una Supply Chain, la cui amministrazione orbita attorno a tre fattori centrali: il flusso di *materiali*, il flusso di *informazioni* e il *tempo di risposta* alla domanda del cliente. Tale struttura è di fatto un sistema di processi interdipendenti, dove le azioni di una parte influenzano quelle di tutte le altre. Il fattore scatenante e principale attore

della rete che agisce al di fuori della catena, controllandola, è la domanda del cliente finale: solo esso è libero di decidere quando effettuare un ordine. Dopo di che, la macchina si deve mettere in moto di conseguenza.

La logistica è un componente di fondamentale importanza all'interno dei una supply chain e ha il compito di gestire due dei flussi chiave:

- il flusso di materiale di beni fisici dai fornitori attraverso i centri di distribuzione fino ai negozi;
- il flusso di informazioni sulla domanda a partire dal cliente finale risalendo per poter regolare l'acquisto dei prodotti e infine il rapporto con i fornitori, in modo che il flusso di materiale possa essere accuratamente pianificato e controllato.

Gestire minuziosamente tali flussi è una parte fondamentale del compito generale del supply chain management, che si occupa dell'intera catena dei processi, compresi la fornitura di materie prime, la produzione, il confezionamento e la distribuzione al cliente finale. Al fine di operare al meglio in questi campi, generalmente ci sono tre attività da ottimizzare:

- *distribuzione*: sia dal punto di vista operativo che di supporto per la gestione dei centri di distribuzione, che per quel che riguarda la distribuzione stessa dei prodotti dai centri dedicati fino ai clienti finali o ai punti di vendita;
- *pianificazione della rete* e della *capacità*: garantendo che i prodotti giusti possano essere acquistati nelle giuste quantità e al momento giusto, ora e in futuro;
- *sviluppo della supply chain*: ovvero apportare migliorie continue ai suoi processi per renderli stabili e sotto controllo, in maniera tale che essa sia efficiente e correttamente strutturata per soddisfare le esigenze logistiche dei flussi materiali e informativi che la coinvolgono.

La logistica può quindi essere vista come parte della sfida globale della supply chain, anche se i termini "logistica" e "supply chain" sono spesso usati in modo intercambiabile. La logistica è in realtà un sottoinsieme che rappresenta solo una parte della gestione della supply chain.

Considerandone la struttura sopra descritta, sarebbe più corretto parlare di “supply network”. Una rappresentazione grafica fedele di una supply chain si presenterebbe dunque nella maniera sotto riportata, con un’azienda principale al centro della rete di relazioni che si creano tra i membri:

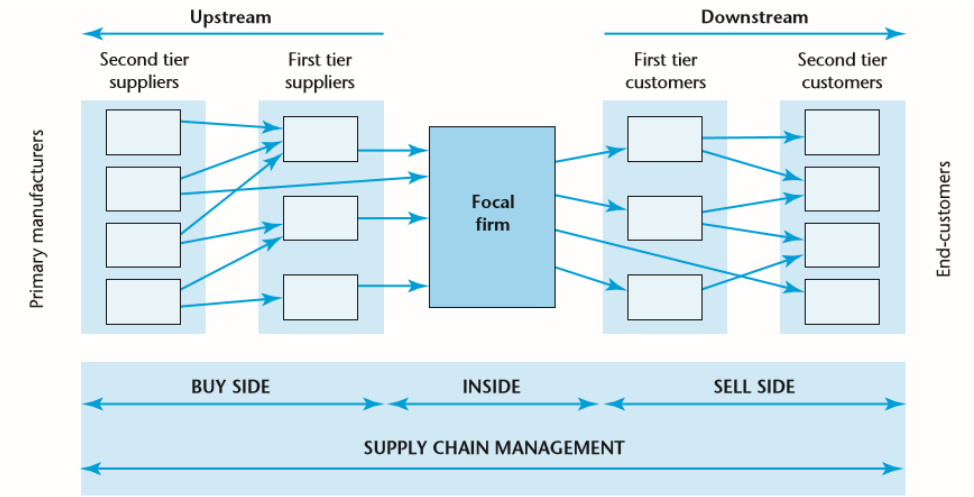


Figura 1-struttura "a rete" delle supply chain odierne

La supply chain si estende dunque anche oltre i confini organizzativi. La "focal firm" è incorporata all'interno della catena stessa, per cui i suoi processi operativi ("inside") devono coordinarsi con altri che ne fanno parte. I materiali si spostano in un flusso da sinistra (a monte, o 'buy side') a destra (a valle, o 'sell side') e le informazioni in senso opposto. In un caso ideale solo il cliente finale (all'estrema destra della catena) è libero di fare ordini quando vuole: dopo di che, il sistema prende il sopravvento e si adatta di conseguenza.

La supply chain si articola su più livelli (*tier*), e il lato della domanda può essere organizzato in gruppi di partner con cui abbiamo a che fare. Ponendo ad esempio come azienda principale della supply chain un centro di produzione di automobili (es. FCA), il lato acquisti si interfaccia con i fornitori di livello 1, o tier 1, dei principali componenti e sottoinsiemi che consegnano direttamente a FCA, mentre i tier 2 si relazionano con i tier 1, ecc. Sul lato delle vendite, FCA si riferisce alle filiali di vendita intrnazionali come tier 1, che a loro volta forniscono ai concessionari principali come tier 2, e così via.

La gestione della supply chain coincide quindi come la gestione "end to end" della rete nella sua interezza, compresa delle relazioni tra i vari nodi che la compongono. Ciò richiede un processo decisionale strategico, un accurato bilanciamento delle scorte e l'integrazione dei sistemi.

La dimensione temporale affronta invece la domanda "Quanto tempo ci vuole per passare dalla supplier al cliente finale?" Cioè, quanto tempo ci vuole per far passare il prodotto attraverso le varie fasi e i vari tier da un'estremità all'altra della supply chain? Il tempo è fondamentale perché misura quanto velocemente una data rete può rispondere alla domanda, adattandosi alle necessità del cliente. Infatti, il concetto di flusso si basa sul tempo.

Il flusso misura la quantità di materiale (in termini di input, come il numero di componenti, tonnellate e litri) che passa attraverso una data rete nell' unità di tempo.

1.2. Supply Chain: criticità e digitalizzazione

Con la globalizzazione, le supply chain coinvolgono oggi migliaia di fornitori provenienti da diverse parti del pianeta. Questo mette in risalto i limiti delle reti e rende fondamentale l'utilizzo di alcune misure volte a ridurne la scala almeno a livello digitale. Ad esempio, **garantire la trasparenza** end-to-end e il tracking delle risorse in tempo reale, può aiutare a contenere il numero delle lacune nella gestione del rischio della supply chain, che a sua volta porta in molti casi a interruzioni, perdite nelle vendite e costi extra.

Inoltre, **prevedere la domanda** di un particolare articolo in modo accurato risulta più complesso. Per creare un modello previsionale adatto si devono infatti tenere in conto molte più variabili, fattore che ne aumenta la complessità, il rischio di trascurare fattori e di conseguenza il rischio di errori nella previsione stessa. D'altro

canto, avendo gli strumenti adatti, con la quantità di informazioni reperibili si possono ottenere modelli di gran lunga più precisi.

Aumentando il numero di attori nella supply chain, una **bassa interconnessione** rischierebbe non solo di vanificare le potenzialità offerte dal mercato globalizzato, ma di aumentare drasticamente i lead time e complicare il sistema di risposta e reazione a eventuali cambiamenti. Questo creerebbe scompensi anche nel processo di ottimizzazione delle scorte e difficolta nello stabilire un prezzo idoneo alla domanda. Inevitabilmente sorgerebbero complicanze nella gestione ottimale dei **magazzini**, specialmente per quel che riguarda inventario della supply chain e logistica dei trasporti. Non sapere con esattezza le tempistiche di consegna dei materiali, specialmente nel caso in cui vi sia un gran numero di componenti in un prodotto, rischia infatti di creare shortage e rallentamenti nella catena di produzione, con conseguenti perdite di tempo.

La digitalizzazione sta però cambiando l'organizzazione del modo in cui le aziende gestiscono le proprie supply chain e i loro processi logistici quotidiani, rendendole adatte ai ritmi e ai volumi odierni. Lo sviluppo delle soluzioni digitalizzate e dell'industria 4.0 ha creato un nuovo sistema economico completo. Inoltre, i clienti richiedono prodotti innovativi, diversificati e ecologici. Questo crea numerose sfide, ma presenta anche l'opportunità di creare soluzioni e pratiche che migliorano le prestazioni e la produttività.

Tre ragioni possono spiegare tale crescita:

1) dal lato dell'**offerta**, la crescita del commercio mondiale, la tendenza alla globalizzazione negli ultimi cinquant'anni si è tradotta in un numero sempre maggiore di prodotti in circolo, che obbligano l'industria a essere efficiente ed efficace per renderli facilmente disponibili al cliente finale.
2) i **progressi tecnologici** consentono l'automazione dei processi e la meccanizzazione dovunque e i magazzini automatizzati sono ormai diffusi in tutto il mondo.
3) infine, per quel che riguarda la **domanda**, l'avvento dell'e-commerce e della distribuzione tramite un numero limitato di canali, ha dato un'ulteriore

spinta al settore, rendendo i magazzini il luogo in cui il virtuale diventa fisico. È infatti possibile uno scambio diretto di informazioni tra punti vendita e centro di distribuzione, con sistemi pull sempre più capaci di rispondere prontamente ai bisogni del cliente. Infatti, come conseguenza della maggior rapidità dei flussi di informazioni, anche l'aspetto fisico deve mantenere standard più alti per soddisfare la domanda.

Tale cambiamento ha portato alla nascita dell'Industria 4.0, intelligente, autogestita e interconnessa.

Il passaggio al digitale consente alle aziende di tracciare l'intera supply chain in tempo reale, scoprendo ad esempio l'esatta posizione delle merci (ordinate, in transito o in un magazzino). Le soluzioni avanzate tracciano facilmente l'inventario combinando gli aggiornamenti dei partner della supply chain con i dati dei dispostivi IoT.

Algoritmi di machine learning possono invece aiutare a prevedere la domanda di un particolare articolo in modo più accurato analizzando i dati, da cui forniscono distribuzioni di probabilità del volume di domanda previsto, invece di un unico numero di previsione. Questo aiuta le imprese a organizzarsi sia sulla base del potenziale di rialzo che sul rischio di ribasso, pianificando di conseguenza.

Di fondamentale importanza per le aziende, al fine di permettere alle informazioni di fluire senza soluzione di continuità tra fornitori, produttori e clienti sono i software per una supply chain digitale completamente integrata, portando la collaborazione al livello successivo. Essendo una piattaforma condivisa, rompe le barriere tra attori e trasforma la pianificazione in un processo continuo, permettendo una maggiore fiducia, un maggior supporto e soluzioni di pianificazione congiunta, soprattutto quando tra aziende non sussiste competizione. Una tale soluzione ridurrebbe anche i lead time attraverso una migliore comunicazione, in quanto i fornitori possono fornire avvertimenti in anticipo, aumentando la reattività di un'azienda a cambiamenti della domanda. Un'altra caratteristica vitale di questa pianificazione a ciclo chiuso è che le decisioni sui prezzi sono integrate con la

pianificazione della domanda e dell'offerta; i prezzi possono essere modificati in base alla domanda prevista, ai livelli delle scorte e alla capacità di rifornimento.

Infine, si potrebbe riscontrare un effettivo miglioramento nelle capacità di gestione del magazzino, grazie anche a sensori in grado di tracciare le merci in tempo reale e di prevedere con precisione il tempo necessario per l'arrivo di una spedizione. Questo assicurerebbe il ritiro e la consegna puntuale delle merci e faciliterebbe di conseguenza l'organizzazione del lavoro.

1.3. DLT e Blockchain: funzionamento base

Le Distributed Ledger Technologies (DLT) permettono di distribuire le transazioni appartenenti ad un unico registro su diversi nodi o dispositivi informatici, ognuno dei quali partecipa individualmente alla rete replicando, convalidando e salvando una copia di queste. Non c'è un'autorità centrale al comando e ogni nodo che procede alla registrazione e alla memorizzazione funziona in modo indipendente.

I vari DLT differiscono tra loro a seconda delle modalità con cui essi sono strutturati e nelle quali avvengono il funzionamento e il controllo dei dati inseriti (ovvero su come si raggiunge il consenso sulla veridicità di essi).

Tra questi, la blockchain si distingue in quanto è l'unica forma che impiega una catena di blocchi di transazioni tra loro criptate e connesse per raggiungere il consenso tra i nodi della rete. I blocchi sono continuamente aggiunti alla catena e i dati al loro interno non possono più essere modificati o cancellati. Dunque, mentre tutte le blockchain sono DLT, non tutti i DLT sono considerabili blockchain.

La blockchain è una delle tecnologie che negli ultimi anni sta avendo un ruolo centrale nella discussione su come integrare efficacemente e con sicurezza il mondo digitale e quello reale.

Il termine Blockchain è stato utilizzato per la prima volta da Satoshi Nakamoto, all' interno del 'White Paper' che descrive il funzionamento della criptovaluta Bitcoin, che è stata l'origine di molte altre applicazioni. Nel settembre del 2015 nove società finanziarie - Goldman Sachs, Barclays, J.P. Morgan e altre - si sono unite per costruire una nuova infrastruttura per i servizi finanziari basata su una blockchain, diventando uno dei nuovi modelli di Fintech, con annunci quasi quotidiani di nuove startup e progetti aziendali e applicazioni anche esterne alla finanza. Ad esempio, ci si è immediatamente resi conto dell'impatto che questa avrebbe potuto avere nell'ambito della logistica e del Supply Chain Management (SCM). Infatti, una delle premesse fatte da Nakamoto, sottolinea come uno dei ruoli principali della blockchain sia quello di creare trasparenza, fattore indispensabile per gli scambi in denaro, ma non solo. Tutti i membri della catena, che diventa garante della verità, sono così certi di avere accesso agli stessi dati.

Tuttavia, in particolare nelle PMI, si ha ancora scarsa conoscenza di questa tecnologia, che è in fase iniziale di sviluppo. Questo può essere in parte spiegato dal fatto che la comprensione del suo funzionamento e dei vantaggi da essa portati non siano di immediata comprensione. Quali sono i vantaggi concreti che la blockchain può portare ad un'azienda? Come si raggiunge la trasparenza? Che ruoli hanno i vari attori che operano all'interno di essa?

Di seguito riporto una definizione del termine che a mio parere descrive in maniera chiara e completa le sue principali caratteristiche:

"La blockchain è una sottofamiglia delle Distributed Ledger Technnologies (DLTs) in cui il registro è strutturato come una catena di blocchi contenenti le transazioni e la cui validazione è affidata a un meccanismo di consenso; questo è distribuito su tutti i nodi della rete nel caso delle blockchain *permissionless* o pubbliche o su tutti i nodi autorizzati a partecipare al processo di validazione delle transazioni da includere nel registro nel caso delle blockchain *permissioned* o private."

Si possono delineare sette caratteristiche indispensabilmente presenti nella blockchain, che sono anche principi di fondamentale importanza per la tecnologia:

- Decentralizzazione

- Trasparenza
- Sicurezza (basata su tecniche crittografiche)
- Immutabilità (del registro)
- Consenso
- Responsabilità (tracciabilità delle transazioni)
- Programmabilità

Infatti, uno dei ruoli fondamentali della tecnologia è quello di poter creare *asset digitali unici*, in un mondo in cui qualsiasi file digitale può essere replicato senza troppe difficoltà. Ad esempio, nel momento in cui scattiamo una foto con il nostro telefono, viene creato un file. Inviando la foto ad un nostro amico, si genera una duplicazione di essa, che potrà a sua volta essere inoltrata ad altre persone. A ciò non ci sono limiti, e ovviamente qualsiasi documento si invii può essere liberamente modificato da chiunque lo riceva. Nel mondo fisico invece, quando avviene il passaggio di un asset tra individui, non vengono create copie e solo uno di essi ne sarà in possesso. Lo stesso avviene nel caso di asset digitali unici, il documento inviato resterà unico e non sarà possibile replicarlo, fattore fondamentale nel caso in cui debba essere passato del valore e che evita situazioni di "double spending". Infatti, duplicare un asset progettato ad esempio per rappresentare una valuta in digitale significa sminuirne valore sino ad annullarlo.

A seconda delle modalità di gestione del consenso e delle logiche di impostazione del registro, si possono ricavare diverse declinazioni del concetto di blockchain, mettendo maggiormente in risalto alcune delle caratteristiche sopra elencate rispetto ad altre:

- *Database di transazioni criptate:* dal momento che è permessa la creazione e gestione di un grande database distribuito per la gestione di transazioni condivisibili tra più nodi di una rete. Le transazioni registrate all' interno del database saranno strutturate in blocchi, a gruppi di dimensione variabile (a seconda del framework). Questi sono tra loro collegati in maniera tale che la modifica di un blocco sia visibile a catena anche in tutti i blocchi precedenti; possono dunque essere modificati solo con l'approvazione (diretta o indiretta) di tutti i nodi della rete. Dunque, una volta creato un

blocco non può neanche essere modificato, e stessa cosa è di conseguenza valida per le transazioni da esso contenute.

- *Successore del "libro mastro" (ledger):* nell' approccio classico, i ledger sono 'centralized' e sottostanti dunque ad un rapporto uno a tanti, con un'autorità centrale alla quale fanno affidamento tutte le altre. Quando si parla invece di ledger decentralizzato si considera una rete all'interno della quale non si fa riferimento ad un'unica autorià centrale ma a tante. In questo caso il rapporto di fiducia è comunque uno a tanti, con un referente che è però più vicino a livello logico. Infine, si arriva al concetto di ledger distribuito (DLT, appunto), dove non esiste più un centro e le relazioni all'intenro della rete si basano sul concetto di fiducia reciproca, costruita grazie al consenso.
- *Registro pubblico immutabile:* le transazioni e gli asset sono infatti archiviati all'interno di una rete peer-to-peer[1]. La stessa "informazione" è dunque presente su tutti i nodi e pertanto diventa immodificabile se non attraverso un'operazione che richiede l'approvazione della maggioranza dei nodi della rete e che in ogni caso non modificherà lo storico di quella stessa informazione. È così permessa e garantita la reale immutabilità dei dati poichè è possibile garantirne e certificarne la storia completa.

[1] Volendone dare una definizione, si può dire che **una rete *peer-to-peer* è un'architettura di calcolo distribuito** nella quale i vari nodi condividono tra di loro parte delle loro risorse informatiche (come la potenza di calcolo della CPU, lo spazio sull'hard disk e la memoria RAM, la larghezza di banda e così via) senza che ci sia bisogno di un nodo centrale (server) che coordini il tutto. I nodi, dunque, non sono gerarchizzati tramite il binomio *client* e *server*, ma **si tratta di nodi paritari** (*peer* in inglese, per l'appunto), in grado di svolgere sia il "ruolo" di cliente, sia quello di servente.

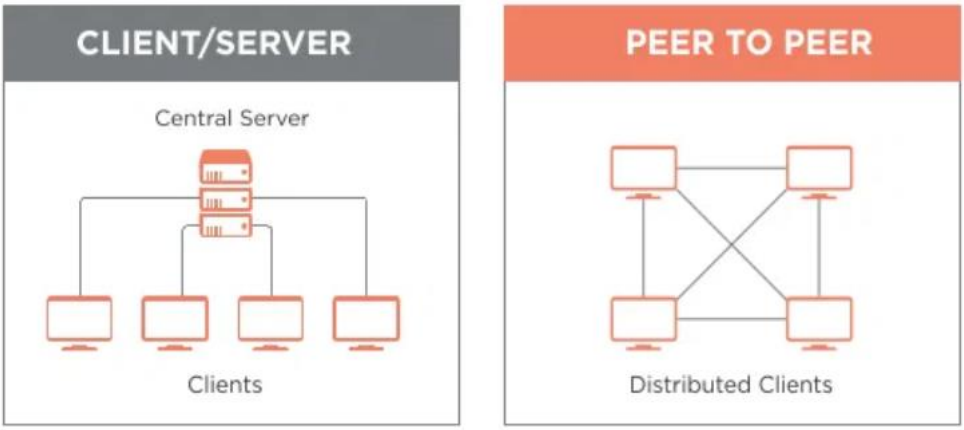


Figura 2 - differenza tra architettura client/server e P2P

Ma come avviene la creazione e la validazione di una transazione all'interno di una blockchain? Di fatto non esiste una risposta unica a questa domanda, dal momento che queste modalità variano a seconda del framework utilizzato, ma si può provare a riassumere dei passaggi generici che accomunano quasi tutte le soluzioni:

1) Qualcuno richiede una transazione tramite a partire da un indirizzo[2] ad esso appartenente.
2) La transazione viene inviata (trasmessa) a tutti i nodi che partecipano ad un determinato network della stessa blockchain.
3) Ogni computer della rete controlla (convalida) la transazione in base ad alcune regole e procedimenti di validazione, stabiliti dai creatori del framework blockchain.
4) Le transazioni convalidate sono immagazzinate in un blocco e sono criptate (hash).
5) Questo blocco diventa parte della blockchain quando altri nodi della rete verificano che questo sia corretto e all'interno di esso non siano presenti transazioni 'maligne'.

[2] Nel caso delle criptovalute, i vari indirizzi appartenenti ad un utente e univocamente identificati da un codice sono contenuti all'interno di un 'wallet', ovvero un portafoglio digitale (e-Wallet) simile ad un conto bancario, che come tale anch'esso possiede un ID proprio e univoco.

6) Avvenuto ciò, la transazione è parte della blockchain e non può essere modificata in alcun modo.

Trattando l'argomento in maniera generica è difficile comprenderne il reale funzionamento e le potenzialità, ma nel capitolo successivo andremo ad esaminare più nel dettaglio ciò che è appena stato spiegato.

1.4. IoT e DLT in una Supply Chain: benefici

L'Internet of Things (IoT) descrive la rete di prodotti incorporati con sensori, software e altre tecnologie allo scopo di connettersi tra loro e scambiare dati con altri dispositivi e sistemi per mezzo di Internet.

La definizione si è evoluta grazie alla convergenza di più tecnologie, come analisi in tempo reale, apprendimento automatico e sistemi embedded. Nel mercato dei consumatori, la tecnologia dell'internet degli oggetti è maggiormente sinonimo di prodotti che rientrano nel concetto di "smart home", con dispositivi ed elettrodomestici che interagiscono all'interno di uno o più ecosistemi comuni, e che possono essere controllati tramite dispositivi associati a tale ecosistema, come smartphone e altoparlanti intelligenti.

L' "Industrial IoT" (IIoT) si riferisce invece all'utilizzo della tecnologia in ambienti industriali, funzionale in particolare modo al controllo di sensori e dispositivi che utilizzano tecnologie cloud. Recentemente, le industrie si stanno concentrando nell'ambito della comunicazione "Machine to Machine" (M2M) per poter permettere ai mezzi che utilizzano di scambiarsi informazioni utili direttamente e per aumentare il livello di automazione e il controllo wireless. L'IIoT è parte della quarta ondata della rivoluzione industriale, o Industria 4.0, e viene adoperato principalmente per:

- Smart production
- Manutenzione preventiva e predittiva
- Reti elettriche intelligenti

- Smart Cities
- Logistica connessa e intelligente
- Supply chain digitali e intelligenti

In quest'ultimo caso, i sistemi IoT vengono spesso abbinati a Distributed Ledger Technologies, utili in particolar modo dal punto di vista della sicurezza che conferiscono ai dati. Infatti, uno dei problemi principali che affligge le reti di oggetti intelligenti, in particolar modo se utilizzate in ambienti industriali, è dato dal fatto che maggiori sono le connessioni maggiore è il numero di potenziali punti deboli che potrebbero essere soggetto di attacchi. Questo rende necessaria l'adozione di pratiche volte a rinforzare l'architettura di tali reti; la cifratura e l'immutabilità che molti framework di blockchain offrono, possono essere il partner perfetto dei sensori all'interno dell'Industria 4.0.

Bisogna però considerare che, a causa del basso livello di digitalizzazione, la maggior parte delle Supply Chain si trova oggi a dover affrontare diversi **problemi** che ne derivano, che possono essere così riassunti:

1. *Gestione inefficiente dei magazzini e degli inventari*, che presentano spesso dei 'visibility gaps' per quel che riguarda la localizzazione e lo stato dei prodotti. Questo si traduce in costi maggiori, carenze di prodotti e clienti insoddisfatti per consegne non avvenuta in tempo.

2. *Processamento delle letters-of-credit* [3] *costoso in termini di tempo e denaro*: le banche sono infatti strettamente legate alle supply chain: le letters-of-credit (o titoli di credito) sono infatti necessarie per garantire la volontà di pagamento, ma implicano processi che richiedono risorse.

[3] *Letters-of-credit*: garanzie di pagamento per i prodotti spediti.

3. *Ritardo nei rimborsi per i beni danneggiati*: derivato dall'impossibilità di tracciare beni danneggiati, questo si rispecchia nel fatto che il capitale impegnato venga ritenuto per periodi di tempo maggiori.

4. *Dipendenza da intermediari:* che devono provare l'autenticità dei documenti, mitigare il rischio, minimizzare i ritardi, e aiutare il business ad evitare sanzioni. Questo però comporta un rallentamento e costi extra nel processo di spedizione.

5. *Costi extra di frodi e furti:* inoltre, il fatto di non essere in grado di assegnare responsabilità per i furti rappresenta un problema, e può risultare in cause, con i costi che ne derivano.

6. *Disposizione e tracking dei beni inefficiente,* con conseguente sfruttamento non ottimale dei container e costi di spedizione più alti.

Tramite l'inserimento di sensori **IoT** si possono ottenere notevoli miglioramenti notevole, a partire dai problemi legati alla visibilità, dove è però offerta solo una soluzione parziale. Infatti, i dati raccolti e immagazzinati in un database centralizzato godono di scarsa fiducia da parte delle altre figure da essi coinvolte. Proprio in questo frangente l'utilizzo di una **blockchain** o più genericamente di un **DLT** può essere d'aiuto (grazie alle caratteristiche esposte nel paragrafo precedente), permettendo:

1. tracciabilità in tempo reale e trasparenza
2. automazione del processo di acquisto
3. eliminazione delle tassazioni sulle letters-of-credit
4. risoluzione più veloce delle dispute
5. minor dipendenza da intermediari
6. riduzione al minimo di furti e frodi
7. migliore utilizzo dello spazio nei trasporti (es. container)

Detto ciò, di sicuro non ogni problema che si presenta all'interno di una supply chain è facilmente risolvibile con l'utilizzo di questa tecnologia, in particolar modo se non appaiata a sensori che facciano da garanti per la veridicità e l'incorruttibilità del dato all'origine.

Nella seguente tabella sono riassunti i punti critici nella gestione delle supply chain che si possono risolvere con l'implementazione di soli sensori IoT e di questi ultimi insieme ad una blockchain:

☹ = non risolto
😐 = parzialmente risolto
☺ = completamente risolto

	Consolidamento del commercio	Trasporti	Rapporti	Spedizioni a magazzino	Distribuzione
Processo tradizionale	☹ Spedizioni importanti non prioritizzate ☹ Processamento delle L/C costoso	☹ Prodotti danneggiati durante la spedizione ☹ Ritardi nelle pratiche assicurative	☹ Dipendenza da intermediari	☹ Frodi e furti di prodotti	☹ Disposizione prodotti inefficiente ☹ Pianificazione delle tratte inefficiente
Con IoT Visibilità in tempo reale	☺ Prioritizzazione delle spedizioni urgenti ☹ Processamento delle L/C costoso	😐 Monitoraggio dei prodotti danneggiati ☹ Ritardi nella verifica delle assicurazioni	☹ Dipendenza da intermediari	😐 Meno frodi e furti	☹ Disposizione prodotti inefficiente ☺ Efficienza grazie a dati traffico in tempo reale
Con IoT e Blockchain Azioni in tempo reale	☺ Tracciabilità in tempo reale ☺ Smart contracts automatizzano L/C	☺ Disputa assicurativa istantanea e risoluzione rapida dei conflitti	☺ Nessuna necessità di intermediari	☺ Frodi rilevate in tempo reale	☺ Disposizione prodotti efficiente che migliora l'utilizzo dei container

Tabella 1 - contributo di blockchain e IoT all'interno di una Supply Chain

1.5. Step e requisiti per l'implementazione

1.5.1. Requisiti

Il requisito principale (e non scontato) perché si possa pensare ad una soluzione che considera l'uso della blockchain, è che ve ne sia realmente il bisogno. Infatti, se

non sono presenti alcuni fattori si può tranquillamente ricorrere all'impiego di altre tecnologie che non posseggano per forze le caratteristiche che sono i punti di forza della blockchain, la quale è immutabile e distribuita. Questo potrebbe infatti rappresentare un costo extra non indispensabile.

Per prendere questa decisione si può far riferimento ad un albero decisionale simile a quello qui presentato.

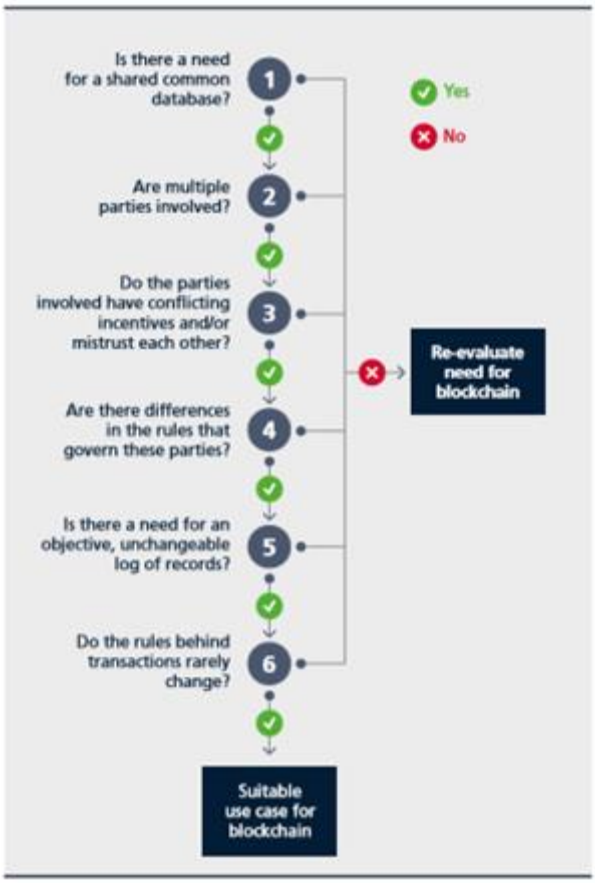


Figura 3-valutazione necessità di una blockchain

Sono inoltre elementi indispensabili affinché sia possibile implementare una blockchain all'interno di una supply chain:

- un ***obiettivo*** chiaro per cui viene implementata la blockchain stessa.

- avere chiari quali siano i punti di ***inizio e fine*** della supply chain.

- definire quali siano i ***dati essenziali*** e per quale motivo siano richiesti.

- vi devono essere sufficienti ***informazioni in relazione alla supply chain e al caso d'uso*** specifico.

- deve esservi una consistente ***integrazione*** all'interno della supply chain, tra i suoi membri, in termini di ***infrastrutture.***

Sono inoltre individuabili dei fattori che determinano quanto i dati riguardanti un prodotto che si muove tra gli attori di una determinata supply chain siano adatti ad essere monitorati con una blockchain. Questi si possono dividere in 'aree di complessità':

a. **Complessità tecnica** di stabilire e mantenere l'identità del prodotto nel tracciare il dato.

b. **Complessità operativa e organizzativa** di costruire le capacità core tra attori e processi per abilitare la tracciabilità.

c. **Complessità di mercato**: data dalla difficoltà di introdurre la tecnologia di tracciamento nel contesto di un dato mercato o regolamento.

d. **Complessità finanziaria:** prettamente nel finanziare la tecnologia e le infrastrutture per il tracciamento.

A seconda di quanto sia facile oltrepassare queste barriere, i possibili casi di applicazione si possono ordinare dal più adatto al meno adatto.

1.5.2. Step per l'implementazione

Innanzitutto, prima di implementare la tecnologia stessa, è importante valutare se questa nella Supply Chain in esame vi siano i prerequisiti necessari affinché questa sia applicabile. Perché questo avvenga bisogna far si che l'infrastruttura di rete presente nelle singole aziende che la compongono sia adatta a supportarne il funzionamento. In particolare, è importante valutare quale tra i componenti è il punto debole; l'efficienza stessa del flusso di informazioni di una supply chain è determinata da quella del suo elemento più debole (collo di bottiglia).
Bisogna quindi valutare anche a che livello sia conveniente iniziare il tracking che, ad esempio, per alcuni tipi di materie prime potrebbe essere complicato.
Facendo ciò si può sviluppare **un'idea** e portarla a una soluzione scalabile (seguendo ad esempio i passi illustrati nella figura di seguito). La fase di progettazione e pianificazione è volta a permettere l'ideazione di un prototipo per la **Proof-of-Concept**. Durante questa fase, le parti interessate dovrebbero impegnarsi in training per imparare tutte le sfumature dell'utilizzo della tecnologia nell'applicazione proposta. Nella **fase pilota**, si passa dunque a testare l'applicazione su piccola scala, valutandone però già l'applicabilità su larga scala. È dunque imperativo includere tutti gli stakeholders, apportando così un cambiamento di prospettiva: si sposta l'attenzione da una visione interna all'implementazione su più parti e al test della soluzione attraverso l'intera rete.
La fase finale di un'implementazione in blockchain comporta la **scalabilità** della soluzione e il raggiungimento di **vantaggi** omnicomprensivi. Ciò richiede una significativa trasformazione dei processi aziendali tra più stakeholder, compresi i partner commerciali e persino i concorrenti. Pertanto, il successo della soluzione dipende in larga misura da:

- adozione e accettazione della tecnologia da parte di tutti i membri della supply chain
- adeguamento delle pratiche aziendali da parte di tutti (che è probabilmente il fattore più critico)
- corretto sfruttamento dell'implementazione della blockchain.

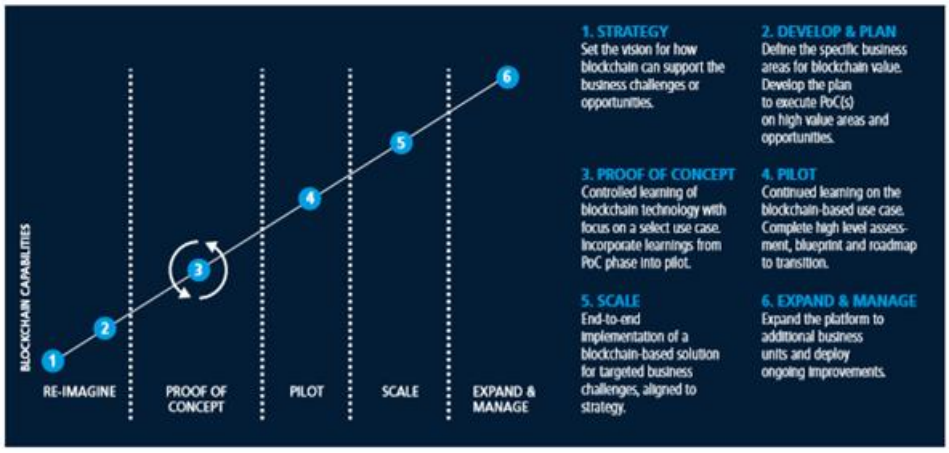


Figura 4-step per l'implementazione di una soluzione blockchain/IoT

2. DLT e Blockchain: tecnologia, funzionamento e potenzialità

La popolarità della Blockchain è storia recente, ma il concetto di DLT è in realtà ben più vecchio, nonostante sia spesso associato a quello di blockchain, creando confusione tra i due. Entrambi si possono considerare metodi per organizzare i record delle transazioni in un database condiviso e distribuito, ma come abbiamo già sottolineato in precedenza DLT è un termine ombrello che comprende diversi tipi di strutture.

Bisogna ricordare che i "distributed ledgers" non devono necessariamente usare blockchain, e le blockchain non devono per forza avere una struttura distribuita (sebbene il loro valore pratico sia messo in discussione senza tale caratteristica).

Tuttavia, nonostante l'avvento dell'informatica, per molto tempo i Ledger sono stati fortemente legati alla stessa **logica centralizzata** che caratterizzava il corrispettivo carta, con qualcuno che, *centralmente*, gestiva l'inserimento e le estrazioni di dati. In questo periodo, il dato ha continuato ad essere concepito e gestito come se fosse analogico.

Nonostante non mancassero le idee e le soluzioni per fare questo grande salto verso una digitalizzazione di molti processi all'interno dei Ledger, mancavano soluzioni in grado di garantire **alti livelli di sicurezza**. L'uso del DLT è stato infatti storicamente ostacolato a causa di un dilemma che divenne noto come il problema dei generali bizantini. Immaginate questo scenario: diversi generali al comando dei rispettivi eserciti sono strategicamente posizionati al di fuori del territorio nemico. I generali devono comunicare con i messaggeri per formare un accordo comune. Ma cosa succede se un generale corrotto cospira contro gli altri per impedire loro di raggiungere un obiettivo comune? Tale problema, descritto da Leslie Lamport, Robert Shostak e Marshall Pease in un loro saggio del 1982, si presenta in maniera analoga quando diversi nodi facenti parte di una stessa rete devono concordare sulla validità di una transazione. Una risposta a questo problema è stata recentemente

fornita dalla blockchain. La tecnologia è basata su una combinazione tra firma digitale e *timestamp*[4]: la prima garantisce l'identificazione certificata del mittente e del destinatario di un qualsiasi tipo di messaggio, il secondo rende possibile la validazione di un insieme di messaggi e la sua comunicazione a tutti i nodi della rete in maniera irreversibile.

Come visto in precedenza, nel caso di Bitcoin tutte le transazioni sono confermate dalla rete nel giro di dieci minuti dal momento della pubblicazione, attraverso il *mining*, dove avviene un confronto tra l'ultimo blocco e la versione più aggiornata della blockchain posseduta dai nodi. Il primo nodo che completa la PoW lo comunica a tutti gli altri, che provvedono alla convalidazione aggiornando la blockchain, per preservare sia l'ordine cronologico che la neutralità della rete.

2.1. Origini: Bitcoin

Come accennato all'interno del capitolo precedente, la Blockchain nasce nell'Ottobre 2008, quando Satoshi Nakamoto[5] pubblica il white paper intitolato "Bitcoin: A Peer-to-Peer Electronic Cash System." Leggendolo ci si è rende conto che ciò che viene descritto all'interno rappresene un punto di svolta importante, nei confronti di tutti i sistemi organizzativi centralizzati presenti al giorno d'oggi, come ad esempio le banche. Sin da quando è stata rilasciata prima unità di bitcoin Cryptocurrency, la Blockchain di Bitcoin ha dato prova di poter funzionare, seppur oggi esistano di fatto esistono relativamente pochi metri di paragone con essa.

Inoltre, Bitcoin svolge il ruolo di "asset politico", grazie a ciò che intrinsecamente esprime riguardo ai sistemi finanziari e monetari, ponendosi a favore di un approccio completamente distribuito, trasparente, immutabile e di conseguenza democratico. Infatti, *Trust*, ovvero fiducia, è la parola chiave per comprendere la

---

[4] Metodo utilizzato nei sistemi operativi Unix e Unix-like per rappresentare il **tempo** come *offset* in secondi rispetto alla mezzanotte (UTC) del 1º gennaio 1970 (detta *epoca*).
[5] Per alcuni **Satoshi Nakamoto** non è un individuo singolo, ma si tratterebbe in realtà di uno pseudonimo volto a nascondere l'identità di un gruppo di sviluppatori altamente competenti.

blockchain sia dal punto di vista del funzionamento che come concetto. In relazione a ciò, il messaggio lanciato da Nakamoto è di questo tipo: "*We define digital coin as a chain of digital signatures*". Sono proprio tali firme ad assolvere il compito di "creare" fiducia tra individui che non si conoscono, senza la necessità di affidarsi a terze parti. Questa si può considerare una delle applicazioni più "raffinate" della teoria dei giochi.

Tale logica di Trust, è anche resa possibile dalla risoluzione di un noto problema matematico conosciuto come "*Problema dei Generali Bizantini*", riuscendo a delineare una formula per il raggiungimento del consenso tra più parti. Tale problema parte dalla necessità di un generale bizantino, durante un assedio, di inviare, attraverso dei messaggeri, l'ordine di attaccare il nemico ad altri luogotenenti che si trovano in postazioni diverse.

Tutti sanno però che tra di loro si trovano uno o più traditori. Come si può dunque avere la certezza che sia stato inviato lo stesso ordine anche agli altri?

È proprio la forza del consenso che mette le radici per la sicurezza della blockchain e riesce a trovare una soluzione a questo tipo di problema.

Al contrario delle più diffuse reti centralizzate, la blockchain si pone dunque come un modello distribuito; ogni nodo ha una propria capacità computazionale, e vede esattamente le stesse informazioni che vedono gli altri, fidandosi del contenuto di queste.

2.2. Modalità

Esistono però diversi tipi di blockchain, e più in generale delle Distributed Ledger Technology, che differiscono a livello macro per la loro apertura ad accogliere nuovi nodi partecipanti; occorre dunque introdurre i concetti di *Permissionless* e *Permissioned* Ledger (blockchain Pubbliche e Private).

2.2.1. Blockchain Pubbliche

Rappresentante per eccellenza di questo tipo di Blockchain è proprio Bitcoin. Tali soluzioni simboleggiano la concezione più 'pura' di blockchain, in quanto sono aperte e non hanno di fatto una "proprietà" o un attore di riferimento.

All'interno di tutti i framework di *Permissioneless Ledger*, è possibile per ogni partecipante contribuire all'aggiornamento dei dati, validandoli, e possedere le copie immutabili di tutte le operazioni, ovvero la loro versione locale della blockchain. Qui, nessuno può impedire che una transazione avvenga, e sarà aggiunta al Ledger solo una volta che ha conquistato il consenso necessario tra tutti i nodi.

Inoltre, tale tipo di ledger potrebbe essere adatto ad un utilizzo globale come database, per immagazzinare documenti che necessitano di essere assolutamente *immutabili* nel tempo (come ad esempio possono essere i contratti di proprietà o i testamenti).

2.2.2. Blockchain Private

Utilizzate comunemente per le soluzioni "business related", come istituzioni, grandi imprese e intere supply chain, i *Permissioned Ledger* possono invece avere una "proprietà". In questo caso l'inserimento di un nuovo dato nella blockchain sarà vincolato ad approvazione, non basata però sulla maggioranza dei partecipanti (come Bitcoin), bensì su di un numero limitato di attori definiti come *Trusted*. Inoltre, a questo dine sono definite speciali regole di accesso e visibilità dei dati. Si introduce così nella blockchain il concetto di Governance per la definizione di un set di regole comportamentali. Tali regole si traducono anche in algoritmi di consenso alternativi alla PoW, che consentono il raggiungimento di performance migliori e maggior rapidità nell'approvare le transazioni, rispetto alle soluzioni *Permissionless*.

In queste ultime sono di fondamentale importanza quattro elementi, che devono essere definiti in maniera chiara:

- Infrastruttura
- Ecosistema
- Applicazioni
- Governance

Infatti, la sicurezza di queste soluzioni è direttamente legata alla capacità di garantire l'impenetrabilità dell'infrastruttura da parte di attori indesiderati. Questo concetto è valido del punto di vista della progettazione Hardware e Software, tanto quanto per ciò che riguarda i valori e le regole che devono essere definite, condivise da tutti e che permettono di delineare l'ecosistema in cui la soluzione viene implementata. Di conseguenza, per far sì che le due cose concordino, gli chi si occupa dello sviluppo tecnologico delle soluzioni di blockchain Private deve lavorare a stretto contatto con i fornitori di infrastrutture e con tutti gli attori che ne faranno parte. Infatti, lo sviluppo deve avvenire in maniera concorde alle logiche di governance e al livello di sviluppo dell'infrastruttura. Dunque, alla base dell'intera soluzione deve esservi un insieme di regole condivise dagli attori ed elaborate in fase di ideazione e progettazione. Tutto ciò a fine di implementare correttamente la sicurezza che necessita la soluzione, ma anche per il raggiungimento degli obiettivi economici delle imprese e delle organizzazioni che saranno chiamate a utilizzarla.

2.3. Funzionamento base

Nel capitolo precedente è stata data un'idea generale di come effettivamente questa tecnologia funzioni, ma senza entrare veramente nel merito è difficile che ciò venga realmente capito. Prendendo ad esempio il funzionamento di Bitcoin, possiamo esaminare tale processo più nel dettaglio. Sebbene ai fini della tesi non siano utilizzate criptovalute, capirne il funzionamento è infatti fondamentale per potersi occupare di applicazioni in altri ambiti. Nel seguente esempio vediamo come si passa dall'intenzione di effettuare un pagamento ad ottenere una transazione verificata dalla rete:

Supponiamo che due utenti, che chiameremo U1 e U2, siano entrambi in possesso di un wallet e vogliano effettuare un pagamento tramite criptovaluta, dove U1 è intenzionato a remunerare U2. Perché questo avvengo devono essere completati con successo i seguenti passi:

- U2 crea un nuovo indirizzo all'interno del suo wallet, dove può ricevere il pagamento di U1. Tale indirizzo è di fatto *chiave pubblica*, alla quale viene associata alla creazione anche una corrispondente *chiave privata*. Quest'ultima viene utilizzata per l'autentificazione tramite firma digitale delle transazioni, che effettua il possessore dell'indirizzo. Quella pubblica, serve invece come riferimento per ricevere transazioni da altri utenti. Tale principio sta alla base della *crittografia asimmetrica,* utilizzata principalmente in due scenari:
    1. quando X invia un messaggio a Y, se X cifra il messaggio con la chiave privata Y potrà decifrarlo con la corrispondente pubblica. Questo meccanismo non mantiene la segretezza del messaggio, ma permette a Y di essere sicuro che il messaggio provenga da X.
    2. Quando Y vuole mandare un messaggio a X, se Y effettua la cifratura con la chiave pubblica di Y, questo potrà essere decifrato esclusivamente da X tramite l'utilizzo della sua chiave privata.

- Inizia ora il processo di pagamento: U1 comunica la volontà di eseguire un trasferimento, e utilizza la chiave privata di un suo indirizzo con bilancio positivo (all'interno del quale possiede dunque una certa quantità di denaro in criptovaluta) per firmare digitalmente la transazione all'indirizzo fornitogli da U2.

- Adesso chiunque nel network veda la transazione può usare l'indirizzo del mittente, U1 (ovvero la chiave privata) per decifrare la transazione e scoprirne il contenuto, e verificando di conseguenza l'identità di chi l'ha eseguita.

- Una volta che la transazione viene inviata inizia il processo di verifica, che coinvolge altri nodi della blockchain, chiamati *miners.* Questi hanno il compito raccogliere un certo numero di transazioni e creare un blocco, tramite un meccanismo di consenso chiamato *proof-of-work (PoW)*[6], e che richiede ad essi di utilizzare parte della potenza computazionale che essi possiedono (e che varia in base alla potenza di CPU, GPU, ecc…) per risolvere un "problema crittografico" per tentativi.

- Il primo miner che completa la PoW si aggiudica una ricompensa in criptovaluta e trasmette a tutti i nodi della rete la sua versione del blocco.

- Una volta che il blocco in cui la transazione di U1 è stata inserita viene confermato dalla rete di miners, U2 riceverà il suo pagamento.

2.3.1. La scalabilità della Rete e i Fork

In alcuni casi può avvenire che due o più nodi della rete individuino la soluzione alla PoW in contemporanea. Se questo succede, i due blocchi vengono condivisi contemporaneamente con la rete ed entrambi hanno lo stesso padre. A livello grafico questo può essere immaginato come un *fork* della catena. Quando il blocco successivo viene creato, l'algoritmo lo collega automaticamente alla catena che presenta la difficoltà complessiva maggiore (e quindi la maggior quantità di PoW cumulata utilizzata per la creazione dei blocchi)

Bitcoin considera la conferma di un blocco come certa e sicura nel momento in cui ad esso sono stati collegati circa 6 figli (per una questione di probabilità), permettendo ad oggi di processare circa una transazione ogni 10 minuti. Questo è però un tempo medio, strettamente correlato ad alcuni parametri come la quantità

---

[6] Il problema crittografico consiste nel tentativo da parte dei nodi di indovinare un numero tale per cui l'hash del blocco combinato con quello del numero da come risultato una stringa che ha un certo numero di caratteri a zero (tale numero identifica la difficoltà della PoW). Tale numero e chiamato *nonce*.

delle transazioni che vengono pubblicate in un dato momento e il numero dei partecipanti alla rete.

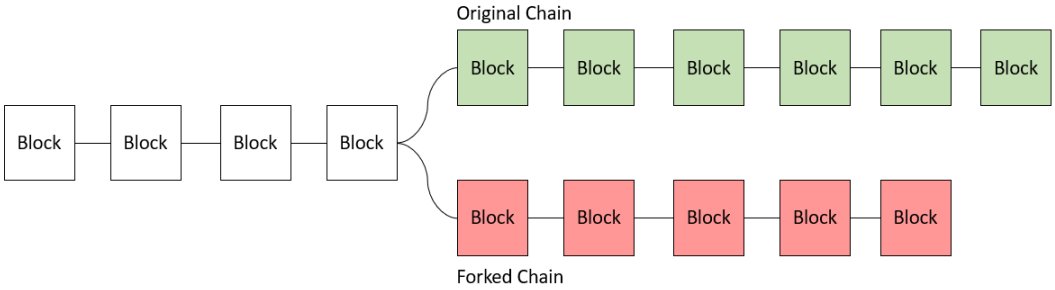


Figura 5-fork nella blockchain

Questo fattore è il principale limite della soluzione di Bitcoin. Infatti, nel 2017, a causa di un aumento del numero di transazioni eseguite, si è registrato un rallentamento nei tempi di consolidamento dei blocchi. Tale rallentamento ha causato un altro tipo di fork, ovvero una vera e propria divisione tra gli sviluppatori che desideravano mantenere la struttura tradizionale del sistema e quelli che invece preferivano aumentare l'output di blocchi e transazioni. Questo ha generato la nascita di diverse blockchain derivanti da Bitcoin (es. Bitcoin Cash e Bitcoin Gold).

Ovviamente, tale soluzione è da considerarsi possibile e naturale, ma bisogna però tenere conto del fatto che frammentare la blockchain in più parti determina un aumento del rischio di accentramento della gestione. Questo intaccherebbe di conseguenza la fiducia tra i nodi, in quanto direttamente proporzionale al loro numero. Perciò è fondamentale che tali situazioni avvengano il meno possibile, e questo fattore di rischio deve essere considerato nel momento in cui si prende la decisione di utilizzare un determinato framework di blockchain.

2.3.2. Possibili minacce alla rete

Come abbiamo detto in precedenza, il meccanismo del consenso protegge la rete da eventuali minacce, rappresentate da nodi malintenzionati. Ma quali sono nel concreto tali avvenimenti pericolosi? Di fatto, un *miner* disonesto potrebbe:

1. Non approvare la trasmissione agli altri nodi della rete di transazioni valide
2. Creare blocchi escludendo le transazioni che sono a loro indesiderate o includendo quelle “maligne”, provando a completare la PoW per primi
3. Tentare di creare un ramo di blocchi corrotti più lungo della catena principale, in maniera tale che questo venga riconosciuto come ramo principale, rendendo quello opposto “orfano”.

Non sarà sicuramente possibile:

3. Creare bitcoin dal nulla, in maniera che vengano accettati da tutti i nodi della rete
4. Effettuare pagamenti per conto di terzi o fingersi loro
5. Effettuare furti di coins da altri account

Per quel che riguarda le transazioni, l’effetto negativo che un singolo nodo può avere è però molto limitato. Se la maggior parte della rete è onesta, rifiuterà di approvare ciò che non riterrà valido.

Con i blocchi il discorso è leggermente diverso. Infatti, se il nodo malintenzionato possiede una potenza computazionale sufficientemente alta, può ritardare la l’approvazione di una transazione rifiutando di includerla nella sua versione dei blocchi. Tuttavia, questa sarà comunque vista dagli altri nodi che la potranno includere nei loro blocchi.

Nel caso peggiore in cui il malfattore possa creare una blockchain di lunghezza maggiore e in tempo minore rispetto a quella creata dal resto della rete, l’algoritmo della blockchain in uso (siamo sempre nel caso di Bitcoin) permetterebbe di escludere il ramo più corto, secondo la "regola della catena più lunga". In questo caso gli sarebbe realmente possibile impedire l’approvazione di una transazione.

Questo escamotage potrebbe essere utilizzato per esempio per effettuare una *doppia spesa,* data una certa quantità di coin a disposizione. Il processo sarebbe tuttavia intricato, ma proviamo a riassumerlo di seguito, mettendoci nei panni di un nodo malintenzionato:

1. In primo luogo, dovrò effettuare due pagamenti con gli stessi bitcoin: uno a un rivenditore online, l'altro ad un secondo indirizzo che si controlla (diverso da quello dal quale sto effettuando il pagamento)
2. Dovrò poi trasmettere solo il pagamento destinato al rivenditore
3. Una volta che il pagamento viene aggiunto in un blocco secondo il processo normale di approvazione, allo il rivenditore invierà la merce che ho pagato
4. Assicuratomi che la merce sia stata inviata, creerò segretamente un ramo di blockchain più lungo, nel quale non sarà incluso il pagamento al rivenditore e al posto del quale sarà inserito il pagamento fittizio rivolto a me stesso
5. Una volta pubblicata la catena maligna, gli altri nodi ignoreranno il blocco onesto con il pagamento al rivenditore, che rimarrà orfano, e secondo regolamento continueranno a collegare i nuovi blocchi al nostro ramo.
6. Il pagamento originale al rivenditore sarà dunque considerato non valido dai nodi onesti, in quanto quei bitcoin verranno visualizzati come già spesi.

Tale processo è riassunto nella seguente immagine:

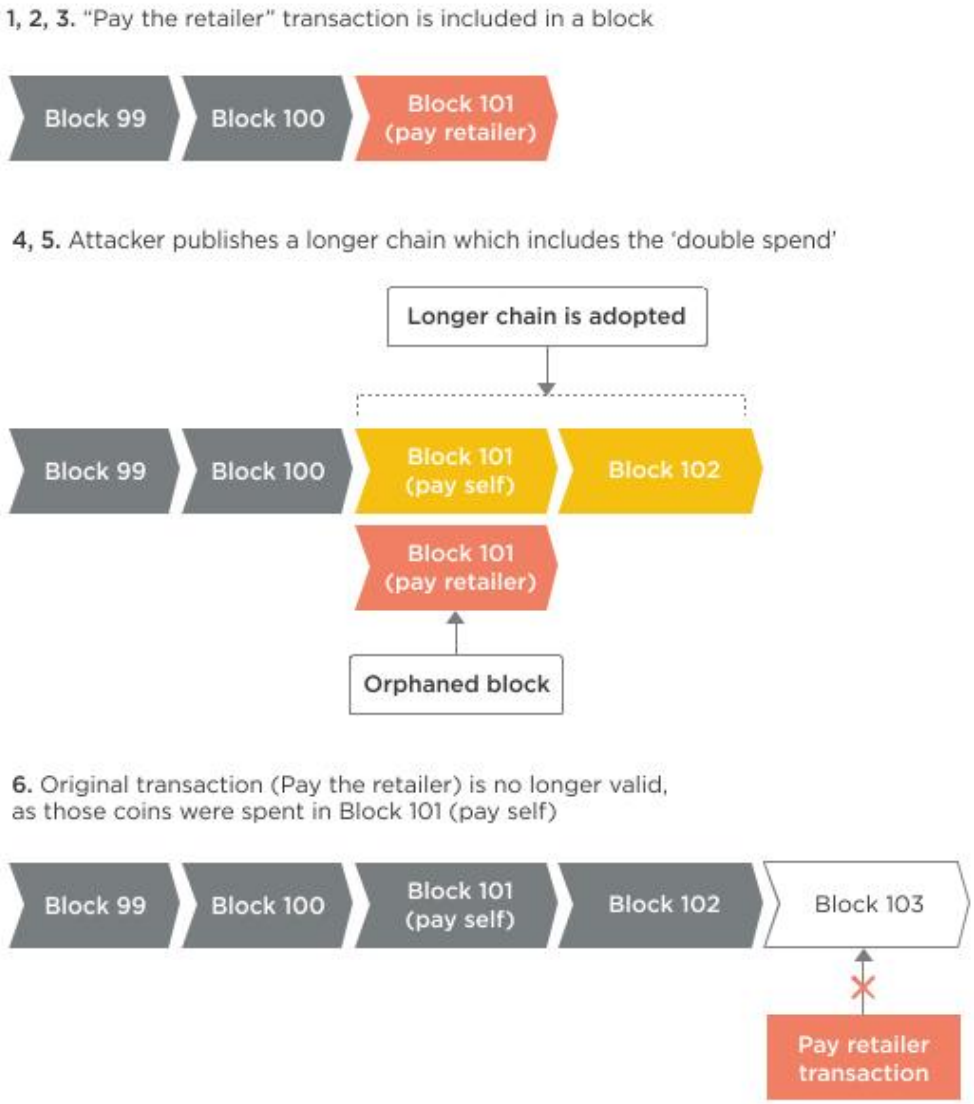


Figura 6-rappresentazione della "doppia spesa"

Questo tipo di attacco è conosciuto con il nome di "doppia spesa", visto che gli stessi bitcoin vengono spesi due volte.

Per rendere difficile, se non praticamente impossibile, la creazione di blocchi a *miners* disonesti, bitcoin agisce rendendo costoso aggiungere nuovi blocchi alla catena, in termini di potenza computazionale. Infatti, un'alta richiesta di potenza

di elaborazione al computer si traduce anche in un alto costo dal punto di vista finanziario (dal momento che ovviamente i computer devono essere acquistati, eseguiti e mantenuti).

La "difficoltà" del mining è infatti legata alla potenza totale di elaborazione della rete. Più computer si uniscono ai blocchi di processo, più raggirare la rete diventa complicato.

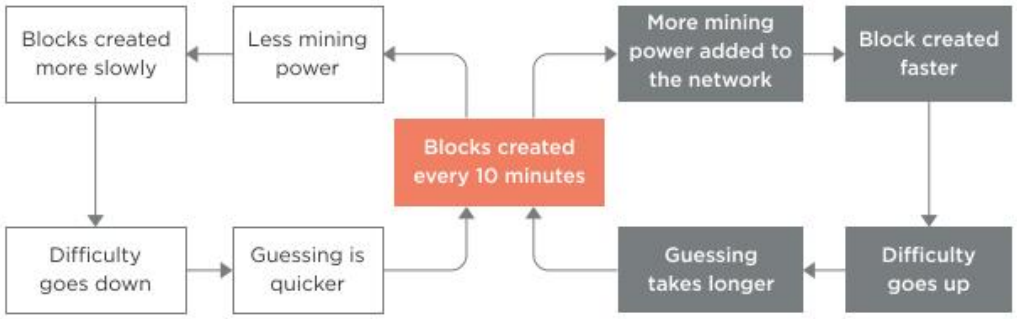


Figura 7-relazione tra potenza computazionale e difficoltà del mining in una rete blockchain

Al fine di regolare questo processo, ogni volta che vengono aggiunti 2.016 blocchi alla blockchain (circa ogni 2 settimane), la rete bitcoin regola la difficoltà della PoW, sulla base della velocità con cui questi sono stati creati.

2.3.3. Algoritmi di Consenso

Come abbiamo visto, i modelli di consenso sono introdotti nelle reti distribuite per approvare le transazioni, assicurandosi che non vi siano tentativi da parte di nodi malintenzionati di 'fregare la rete'. Questi metodi devono essere dunque inequivocabili, o meglio, devono ridurre al minimo la probabilità che quanto descritto avvenga.

I metodi di consenso più usuali si basano sull'assunzione che il maggior investitore non attaccherebbe il sistema stesso sul quale sta investendo, svalutando i suoi ricavi (in caso di PoW) o il suo patrimonio (per la PoS). Tali algoritmi di consenso non sono però gli unici utilizzati nell'ampio numero di framework di blockchain che

esistono oggi. Per cercare di avere una visuale su quali siano i principali e tra questi, ho di seguito riportato i loro nomi e il loro funzionamento:

- ***Proof-of-Work (PoW)***

Ogni nodo deve indovinare un numero tale per cui l'hash delle transazioni combinato con questo dia come risultato una stringa che possiede un certo numero di caratteri equivalente a zero (*nonce*): il numero di zeri è nominato difficoltà della PoW. Dato che un hash crea una stringa crittografata di lunghezza fissa e che non è possibile ricostruire i dati a partire da questo, la combinazione numero e transazioni può essere indovinata solo mediante un certo numero di combinazioni casuali. Di conseguenza, per ogni miner la probabilità di risolvere il puzzle e tanto più alta quanto maggiore è la potenza di calcolo, che si traduce nella capacità del nodo di eseguire un numero più alto di tentativi in un minor tempo. Questo algoritmo causa dunque consumi ingenti in termini di risorse di calcolo, e dunque di energia elettrica.

- ***Proof-of-Stake (PoS)***

È un'alternativa che permette un ottimo risparmio energetico rispetto alla PoW. Invece di chiedere agli utenti di trovare un nonce in uno spazio illimitato, la PoS richiede che le persone dimostrino la quantità di valuta che possiedono, perché si ritiene che posseggono un maggior capitale siano meno propensi ad attaccare la rete stessa sulla quale investono. Tale soluzione è però considerata "ingiusta", perché la persona più ricca è destinata a essere dominante nella rete. Di conseguenza, molte soluzioni vengono proposte sostituendo le dimensioni della 'Stake' con altri fattori, per decidere chi minerà il prossimo blocco. In particolare, Blackcoin (Vasin, 2014) utilizza un metodo randomizzato, eseguendo la selezione tramite un algoritmo che considera la dimensione del conto e l'intervallo di tempo che è passato dall'ultima transazione eseguita dal nodo (*età del coin*) in combinazione, dando più peso a chi

esegue muove valuta con più frequenza. Peercoin (King e Nadal, 2012) favorisce invece la selezione basata sull'età del coin.

- ***Proof-of-Authority***

La Proof of Authority (PoA) si basa invece sul concetto di reputazione, facendo uso del valore delle identità dei partecipanti alla rete. Ciò significa che i chi si propone per validare un blocco mette in gioco la propria reputazione tra i nodi della blockchain, invece che un quantitativo in coin. In altre, le blockchain che utilizzano la PoA sono sorrette da nodi riconosciuti come entità affidabili. Dal momento che questo modello si basa su un numero limitato di miner, è altamente scalabile. I blocchi e le transazioni sono approvati da partecipanti pre-approvati, che svolgono le funzioni di moderatori del sistema.
Per quanto riguarda le Supply Chain, ad esempio, la PoA è ritenuta una soluzione efficace, permettendo un elevato numero di transazioni eseguibili per secondo.
Il modello Proof of Authority consente alle imprese di mantenere la propria privacy e allo stesso tempo avvalersi di alcuni vantaggi della tecnologia blockchain. Microsoft Azure è un esempio di compagnia che applica la PoA. Nonostante ciò, il modello è anche criticato in quanto presuppone l'esistenza a priori di fiducia nei confronti di uno dei player della rete, fattore che dovrebbe essere conferito dalla blockchain stessa e che di fatto crea una sorta di centralizzazione e dipendenza da esso della rete.

- ***Proof-of-Elapsed-Time***

In questo metodo inventato da Intel nel 2016, ogni nodo partecipante alla rete deve attendere un periodo di tempo casuale, e il primo che termina l'attesa designata, acquisisce il diritto di validare nuovo blocco. Durante il tempo di attesa casuale, il nodo non può eseguire la validazione. Quello che 'si sveglia' per primo - cioè quello con il tempo di attesa più breve - inserisce un nuovo blocco nella blockchain, trasmettendo le informazioni necessarie a tutta la rete. Lo stesso processo si ripete

successivamente per la creazione degli altri blocchi. Questo meccanismo ha lo scopo di garantire due importanti fattori. In primo luogo, che il tempo di attesa sia realmente casuale, in secondo luogo, che il vincitore abbia effettivamente atteso l'intervallo previsto. Ciò permette un moderato utilizzo delle risorse e di energia, mantenendo il processo più efficiente secondo un sistema casuale e equo.

- ***Proof-of-Importance***

Gli account saranno valutati sulla base di un punteggio di importanza. Quelli con una valutazione maggiore avranno una più probabilità di essere scelti per creare un blocco. Così facendo, il sistema considera il sostegno che ognuno apporta alla rete (diversamente, ad esempio, dalla Proof-of-Stake). Per fare ciò vengono considerati tre fattori:

Vesting

- Per essere considerati è necessario possedere un minimo di 10.000 coin.
- Più alto è il numero di coin su cui si investe, più alto è il punteggio di importanza del conto.
- Si contano solo le monete che sono state in un conto per un determinato numero di giorni.

Transaction Partners

- Vengono premiati gli utenti che effettuano transazioni con altri conti sulla rete.
- Gli utenti non possono truffare la rete facendo trading a ripetizione, in quanto l'algoritmo contabilizza solo i trasferimenti netti nel tempo.

Numero e dimensione delle transazioni negli ultimi 30 giorni

- Ogni transazione (al di sopra di una dimensione minima) contribuisce al punteggio per la PoI.
- Le transazioni più grandi e più frequenti hanno un impatto maggiore sul punteggio.

Tale sistema è strettamente legato allo scambio di valuta, e le transazioni a valore zero sono penalizzate.

- ***Practical Byzantine Fault Tolerance (pBFT)***

La Byzantine Fault Tolerance è di fatto la traduzione della capacità di una rete di computer distribuita di funzionare come desiderato e di raggiungere correttamente un consenso sufficiente nonostante le componenti dannose (nodi) del sistema falliscano o propaghino informazioni errate ad altri. L'obiettivo è quello di attenuare l'influenza che eventuali nodi maligni hanno sul corretto funzionamento della rete e sul consenso che viene invece raggiunto dai nodi onesti del sistema.
In sostanza, tutti i nodi del modello pBFT sono ordinati in una sequenza in cui un nodo è il nodo primario (leader) e gli altri sono chiamati nodi di backup. L'accordo sullo stato del sistema attraverso una maggioranza. I nodi comunicano tra loro a ripetizione, dovendo da un lato dimostrare che i messaggi provengono da uno specifico nodo peer, ma anche verificando che il messaggio non sia stato modificato durante la trasmissione.
Affinché il modello pBFT funzioni, l'ipotesi è che la quantità di nodi maligni nella rete non possa contemporaneamente eguagliare o superare ⅓ dei nodi complessivi del sistema in una data finestra di vulnerabilità. Più nodi nel sistema sono presenti, più è matematicamente improbabile raggiungere tale numero, e dunque più il sistema sarà efficiente.
Ogni round è determinato da 4 fasi:

1. Un client invia una richiesta al nodo leader chiedendo l'esecuzione di una determinata operazione.
2. Il nodo leader invia la richiesta ai nodi di backup tramite diversi canali.
3. I nodi eseguono la richiesta e poi inviano una risposta al client.
4. Il client attende f + 1 (f rappresenta il numero massimo di nodi che possono essere difettosi) risposte da nodi diversi con lo stesso risultato. Questo sarà dato per valido e sarà considerato come il risultato dell'operazione.

Il nodo leader viene cambiato seguendo uno schema round robin durante ogni turno. Ciò può anche essere sostituito con un protocollo chiamato view change se è passato un determinato periodo di tempo senza che il nodo leader processi una richiesta. Una maggioranza netta di nodi onesti può inoltre targare un nodo leader come difettoso e sostituirlo con il leader successivo in linea.

Questo ha due vantaggi principali: in primo luogo, si può raggiungere il consenso può in maniera rapida ed efficiente. Secondo, la fiducia è completamente dissociata delle risorse possedute da un nodo. Ciò significa che un piccolo user ha tanto potere quanto una grande e ricca organizzazione, aiutando a mantenere entrambi onesti.

- ***Federated Byzantine Agreement***

L'aspetto negativo del pBFT è che si richiede che ogni nodo sia coinvolto in ogni transazione per raggiungere il quorum. Questo è come far sapere a ogni sindaco delle città di uno stato che si sta per piantare un albero nel proprio cortile e far sì che la maggior parte di loro sia d'accordo. Tale procedimento potrebbe essere inutile e rappresentare dunque uno spreco di tempo e risorse.
Nel sistema FBA, ogni nodo può decidere di chi fidarsi e far parte del proprio gruppo decisionale, a cui ci riferiamo come 'fetta di quorum' o 'quorum slice'. Seguendo l'esempio precedente, basta far sapere ai vostri vicini (la vostra fetta di quorum) che volete piantare un albero nel vostro cortile e farli accettare.
Inoltre, l'accordo a livello di sistema viene raggiunto quando tutti i quorum slice comunicano le decisioni e transazioni da loro approvate. Per continuare con l'esempio dell'albero, dopo che voi e i vostri vicini avete accettato di piantare l'albero nel vostro cortile, lo portate al consiglio comunale locale, che a sua volta legittima l'azione per il resto della città.

- ***Istanbul BFT***

Anche questo algoritmo si pone il compito di migliorare il pBFT. Utilizza un consenso trifase, composto da: Pre-Prepare, Prepare e Commit. Il sistema può tollerare F nodi difettosi in una rete di N validator, dove N = 3F + 1, come descritto alla base della Bizantine Fault Tolerance. Prima di ogni round, si sceglie un validator come proposer, seguendo un metodo casuale round-robin. Il proposer creerà quindi una proposta di blocco, che trasmetterà in status 'Pre-Prepare'. Una volta ricevuto il messaggio Pre-Prepare dal proposer, i validator ne aggiornano lo stato a 'Pre-Prepared' e poi trasmettono il messaggio. Qui ci si assicura che tutti i validatori stiano lavorando sulla stessa sequenza e sullo stesso round. Una volta ricevuti 2F + 1 messaggi Prepared, il validatore entra nello stato di Prepared e poi trasmette il messaggio Commit, per informare i suoi pari che accetta il blocco proposto e che sta per inserirlo nella catena. Infine, i validators aspettano che 2F + 1 dei messaggi Commit entrino nello stato di Commited e poi inseriscono anche loro il blocco nella catena.

- ***Tangle***

Una Blockchain è una catena sequenziale di blocchi in cui ogni blocco fa riferimento al suo predecessore cronologico, simile ad una lista collegata. I blocchi contengono transazioni multiple, e sono aggiunti in intervalli di tempo più o meno regolari e discreti. Nel Tangle ogni transazione (invece che un blocco di transazioni) fa riferimento a due transazioni precedentemente pubblicate, formando non una lista collegata, ma una complessa struttura web nota in matematica come Directed Acyclic Graph, o DAG.
La realizzazione di una transazione può essere semplificata in 4 fasi:

- *Firma degli ingressi e/o costruzione di un messaggio*: Nel tangle ci sono due tipi di transazioni: transazioni in cui si trasferiscono token (e quindi è necessaria una firma digitale per dimostrare la proprietà di esso), e

transazioni a "valore zero" che trasmettono semplicemente un messaggio o dei dati (e quindi non è necessariamente necessaria una firma digitale).

- *Selezione e convalida dei suggerimenti*: Nel Tangle un tip è una transazione che non è stata convalidata in precedenza da altre transazioni. La selezione del tip è un processo in cui due tip vengono selezionati a caso utilizzando un algoritmo specifico. Una volta selezionati, i due tips devono essere convalidati per verificare che le loro due rispettive storie (il 'sub-tangle cone' a cui fanno riferimento queste due rispettive transazioni) siano coerenti - il che significa che non ci siano doppie spese o altre forme di imbroglio.
- *PoW:* Una volta che i suggerimenti sono selezionati e convalidati per la coerenza, è necessaria una piccola quantità di Proof-of-Work dove alcune risorse computazionali devono essere spese per trovare la risposta ad un semplice puzzle crittografico.
- *Trasmissione*: Dopo che tutti questi 3 passaggi sono stati completati, la transazione (o le transazioni) possono essere trasmesse ai nodi vicini nella rete peer-to-peer che passeranno le informazioni ai loro vicini e così via.

2.4. Funzionalità

2.4.1. Tokenizzazione

Un token è un asset digitale che può essere memorizzato all'interno di una blockchain e scambiato senza intermediari, che viene usato per tenere traccia di una qualche forma di diritto: la proprietà di un prodotto, l'accesso a un servizio, la ricezione di un pagamento, e altri tipi di beni. Esso infatti è in grado di conferire un diritto di proprietà ad un soggetto sull'insieme delle stesse informazioni che sono registrate. Dunque, al passaggio di un token tra due individui corrisponde il passaggio di un qualche tipo di proprietà.

Ad esso posso assegnare metadati, tra cui ad esempio provenienza, certificazioni o localizzazione di un prodotto. Nel caso questo venga utilizzato come 'digital twin'

di un prodotto, può essere anche utilizzato ad esempio in relazione un codice QR univoco posizionato sul corrispettivo reale.

Per capire come funziona un token, pensiamo ad esso come il corrispondente di un 'gettone' che viene comunemente accettato per effettuare delle transazioni; questo in quanto si è sicuri che una volta effettuato lo scambio qualcun altro lo accetterà a sua volta, potendolo reimmettere nel mercato.

Non bisogna però confondere i token con le Criptovalute come Bitcoin, anche se esistono dei token che vengono utilizzati proprio come moneta. Il loro valore può essere ad esempio ancorato a quello delle valute tradizionali. Questi token si chiamano *stablecoin* e offrono la stesse versatilità e programmabilità delle criptovalute, ma con le garanzie e la stabilità delle monete tradizionali.

Tuttavia, lo scopo dei token non è solo quello di creare monete digitali. Possiamo dividere le applicazioni dei token su Blockchain in due grandi famiglie: i *fungible token* da un lato, legati appunto all'idea di criptovalute, e i *non-fungible token*, dall'altro, legati a oggetti ed entità.

Ciascuno di essi avrà degli attributi, come un codice di identificazione, che li rende unici. Come anticipato, con questi token si può rappresentare qualsiasi tipo di bene, sia digitale che fisico.

I *non-fungible token* sono spesso utilizzati per: gestione dell'identità digitale, progetti di tracciabilità (in questo caso parliamo più correttamente di digital twin) e automazione dei processi della supply chain, voto elettronico, collezionismo (come nella maggior parte dei giochi online).

Inoltre, i token possono anche essere suddivisi in 3 classi, sulla base del tipo di diritto che conferiscono al proprietario di esso:

- *Classe 1:* il token è di fatto analogo ad un coin. Non conferisce diritti ulteriori rispetto a quelli correlati alla proprietà del token o all'esistenza di un determinato soggetto/oggetto. La proprietà del token stesso può essere quasi in tutti i casi trasferita tramite la blockchain di riferimento.

- *Classe 2*: in questo caso si tratta di token che sono in grado di conferire ai proprietari dei diritti, esercitabili nei confronti del soggetto che li ha generati o di terzi. Questi possono di fatto rappresentare il sostituto alle *letters-of-credit* (titoli di credito, ossia i "documenti" che conferiscono al possessore "diritto alla prestazione in esso indicata verso presentazione del titolo").

  Ad esempio:

  1) *Token per smart contracts,* che conferiscono il diritto a ricevere dei pagamenti futuri, che vengono eseguiti in automatico sulla base di determinate condizioni definite a priori;
  2) *Token come asset*: in questo caso il token rappresenta una sorta di diritto di proprietà di un determinato asset (materiale o immateriale), ad esempio quote di una data entità.
  3) *Token per la gestione di prestazioni*, come l'accesso ad alcune infrastrutture informatiche o a servizi.
- *Classe 3*: in questo caso si parla di token a funzione mista, rappresentando una proprietà ma conferiscono anche diritti diversi (un esempio è il diritto di voto). Come per i token di classe 1, non conferiscono diritti ulteriori esercitabili su terzi.

2.4.2. Smart Contracts

Il concetto di smart contract è stato inventato da Nick Szabo negli anni '90 nel suo saggio intitolato "Forming and Securing Relationships on Public Networks" (anche se il concetto di "contratto intelligente" ad ancora prima). In questo documento, egli ha immaginato le regole del contratto codificate come codice informatico. L'utilizzo che ne venne fatto fu inizialmente relativo all'attivazione o disattivazione di una licenza software, al verificarsi di alcune semplici condizioni, e ne cessava il funzionamento alla data di scadenza del contratto.

La differenza tra un contratto normale ed uno Smart Contract è che il primo ha bisogno di un supporto legale sia per la sua stesura che per la verifica e l'attivazione, mentre il secondo richiede supporto solo in fase di stesura. Infatti, questo non è altro

che la “trasposizione” per la verifica automatica delle condizioni che sono presenti al suo interno e per l’esecuzione di azioni che ne conseguono.

Nel momento della stesura, è fondamentale definire in modo estremamente preciso le *fonti* di dati alle quali il contratto deve fare affidamento, le quali devono essere considerate affidabili da tutte le parti che vi partecipano. Dunque, uno Smart Contract è per definizione imparziale e, se gli input sono gli stessi i risultati saranno identici. Questo punto di certezza sposta però sul codice, sulla programmazione e sullo sviluppo il potere decisionale.

Ai contraenti spetta il compito di definire condizioni, clausole, modalità e regole di controllo e azione, ma una volta che il loro contratto è diventato codice e dunque uno Smart Contract e i contraenti lo accettano ecco che gli effetti non dipendono più dalla loro volontà.

In un contesto cui come detto prima l’esecuzione del contratto è imparziale, grazie al codice, la blockchain è fondamentale per garantire quel “trust” nel codice stesso, e la sua immutabilità nel tempo.

Ad esempio, un contratto intelligente poteva essere programmato per il seguente accordo:

Se il volo F è in ritardo di oltre 3 ore, pagare l’importo dell’assicurazione A sul conto di Alice.

“Il volo è in ritardo" rappresenta una condizione del contratto e "pagare l'importo dell'assicurazione A ad Alice" è una conseguenza di tale condizione.

Le conseguenze dell'esecuzione di un contratto intelligente dipendono da una serie di condizioni, che noi chiamiamo “stato”.

Lo stato di un contratto smart può includere, ad esempio:

Volo F: Puntuale

Il conto di Alice: 0

Conto contrattuale intelligente: 1000

Importo di copertura: 100

Pagato: Falso

Lo stato di un contratto intelligente viene pubblicato su un DLT. Quando si verifica un aggiornamento delle condizioni, la rete raggiunge un consenso su di esso e una nuova transazione viene allegata alla rete per aggiornare lo stato del contratto:

Volo F: Ritardo (un riferimento ad un'altra transazione con evidenza del ritardo)

Il conto di Alice: 100

Conto contrattuale intelligente: 900

Importo di copertura: 100

Pagato: Vero

I contratti di questo tipo lasciano una traccia di controllo immutabile sul registro e la loro automazione fa risparmiare tempo e denaro alle parti coinvolte. Il concetto è semplice e può essere applicato in quasi tutti i settori industriali in una varietà di interessanti casi d'uso, dalla tracciabilità delle merci lungo tutta la supply chain allo scambio della proprietà di azioni e obbligazioni.

O ancora, le assicurazioni per autoveicoli, sulla base dei dati riguardanti il comportamento alla guida di un individuo acquisiti misurazioni di dispositivi IoT

(es. superamento dei limiti di velocità), attivano o disattivano clausole di vantaggio o svantaggio.

Tuttavia, i contratti intelligenti sono una tecnologia emergente. Rimane aperta la questione di come si inseriranno nei quadri giuridici esistenti e c'è del lavoro da fare prima che raggiungano il loro pieno potenziale.

Esistono più tipi di smart contracts:

1) *On-chain:* come quelli su Ethereum, fanno parte del protocollo di base. Ciò significa che vengono eseguiti e convalidati da tutti i nodi della rete.
    - Vantaggi: La sicurezza dei contratti è proporzionale alle dimensioni della rete e possono trasferire i token dal loro conto senza che sia necessario fornire una firma.
    - Svantaggi: sono poco scalabili perché i loro programmi devono essere eseguiti da tutti i nodi, sono soggetti a commissioni per la transazione, volatili come il prezzo stesso del token sottostante.
2) *Off-chain:* vengono eseguiti al di fuori del framework in utilizzo. Solo un sottoinsieme di nodi, chiamato comitato, ha bisogno di eseguirli per raggiungere il consenso.
    - Vantaggi: non gravano sul resto della rete, il costo medio di una transazione è basso e prevedibile e il livello di decentramento necessario di un contratto intelligente può essere adattato ad ogni caso d'uso.
    - Svantaggi: per trasferire i token, gli smart contract devono firmare le transazioni per dimostrare di avere accesso all'indirizzo del conto, il decentramento dei contratti smart dipende dalle dimensioni del comitato, dai membri del comitato e dall'entità che lo costituisce.

2.5. Casi d'uso

a. Industria Alimentare

A partire proprio da quest'ultima classificazione, riporto un caso ipotetico di applicazione della blockchain nel mercato alimentare, descritto da Accenture in uno studio del 2018. La classificazione di seguito mostra come il tonno proveniente dall'Indonesia sia l'alimento più portato (tra quelli presi in esame) ad essere abbinato ad una blockchain per il suo tracciamento all'interno della supply chain:

Figura 8-classificzaione prodotti "food" in base a quanto sono adatti al tracking con blockchain (secondo Accenture)

Infatti, il tonno dell'Indonesia contribuisce per il 17% alla fornitura mondiale di tonno e per il 27% al tonno fresco e congelato degli Stati Uniti; tuttavia, la pesca non autorizzata del tonno in sta portando al sovrasfruttamento delle risorse marine nelle acque circostanti. Dunque, gli attori normativi indonesiani e i principali fornitori danno la priorità alla sua tracciabilità in Indonesia. Utilizzando la

blockchain, gli attori della filiera del tonno possono aumentare la velocità con cui il prodotto percorre l'intera supply chain, superando le lacune operative e di mercato e consegnando ai consumatori tonno fresco/congelato pescato in modo sostenibile con meno rischi per la salute e la sicurezza e diminuendo i casi in cui il prodotto deteriora a causa di falle nella conservazione di quest'ultimo.

Nell'attuale supply chain del tonno pescato in Indonesia sono presenti diverse sfide in materia di tracciabilità. Questo infatti si muove attraverso diversi attori della supply chain, dal punto di cattura al punto di vendita al consumatore. Inoltre, vi sono diverse sfide anche dal punto di vista tecnico e operativo, tra cui la mancanza di una corretta separazione dei prodotti, dati non disponibili, inaffidabili e/o non standardizzati e una limitata capacità tecnologica delle aziende Indonesiane.

Dei quattro prodotti sopra riportati, il tonno è risultato essere il più adatto. Con un certo impegno tecnico e operativo, che può essere incentivato attraverso opportunità di mercato e finanziarie che si creerebbero all'interno della supply chain, una soluzione di tracciabilità end-to-end basata su blockchain è dunque considerata applicabile.

I vantaggi e i fattori che apportano valore usando la blockchain potrebbero includere una condivisione dei dati più agile e una maggiore fiducia nei dati raccolti (ad esempio, per sapere se il tonno è stato pescato in acque legali), il miglioramento del marchio aziendale per i prodotti conformi alle normative e un motivo più chiaro per investire in tecnologie abilitanti come l'etichettatura RFID che potrebbero migliorare l'automazione e semplificare le operazioni (e che trovano giustificazione maggiore in prodotti di valore più elevato, come ad esempio il tonno).

b. Logistica

La blockchain ha il potenziale per aumentare l'efficienza e facilitare risoluzione delle controversie nel settore della logistica. Man mano che i documenti digitalizzati e i dati sulle spedizioni in tempo reale vengono incorporati nei sistemi basati su blockchain. Tali informazioni possono essere utilizzate per consentire

l’uso di Smart Contract, che possono automatizzare i processi commerciali di vendita nel momento in cui vengono soddisfatte le condizioni concordate.
Una delle prime startup a perseguire tali applicazioni contrattuali nel settore della logistica è stata ShipChain, che ha progettato un sistema completo basato su blockchain per tracciare e rintracciare un prodotto dal momento in cui lascia la fabbrica fino alla consegna finale a domicilio. Tutte le informazioni rilevanti della supply chain sono registrate in un database immutabile basato su blockchain che può eseguire smart contracts una volta che le condizioni sono state soddisfatte (per esempio, non appena l'autista trasmette la conferma di avvenuta consegna). Un elemento chiave per automatizzare il processo di liquidazione è rappresentato dalla valuta digitale di ShipChain, chiamata "SHIP token". I partecipanti alla piattaforma ShipChain acquistano questi gettoni per pagare il trasporto e regolare le transazioni sulla piattaforma.
La blockchain in combinazione con l’IoT nel settore della logistica consentirà in futuro contratti logistici ancora più intelligenti. Ad esempio, al momento della consegna, un pallet collegato in questo modo sarà in grado di trasmettere automaticamente al sistema una conferma e l'orario di consegna, nonché lo stato della merce. Il sistema può quindi verificare automaticamente la consegna, controllare se la merce è stata consegnata secondo le condizioni concordate (ad es. temperatura, umidità, inclinazione) e rilasciare i pagamenti corretti alle parti interessate, aumentando notevolmente l'efficienza e l'integrità del processo. Blockchain e IoT trovano un’altra applicazione nell’ automatizzare i pagamenti da macchina a macchina (ad esempio, macchine collegate che negoziano ed definiscono il prezzo in base ai servizi di logistica svolti).
Infine, si può pensare di utilizzare Smart Contracts nel settore della logistica digitalizzando delle lettere di credito, processo che attualmente tende a richiedere da qualche giorno a settimane.

3. Valutazione idoneità del framework DLT

Nella tabella seguente sono riportati alcuni degli algoritmi di consenso considerati più adatti per l'applicazione con IoT dalla letteratura, sulla base di alcune caratteristiche rilevanti (quelli considerati poco adeguati sono stati omessi dall'analisi). In seguito per ognuna delle caratteristiche viene valutato l'impatto che essa può avere nel contesto in esame.

Consensus Method	Accessibility	Decentra-lization	Scalability	Throughput	Latency	Adversary tolerance	Computing Overhead	Network Overhead	Storage Overhead	IoT suitability
PoET	Private P. or PL.	Medium	High	High	Low	N/A	Low	Low	High	High
PoS	Public P. or PL.	High	High	Low	Medium	<51% stakes	Medium	Low	High	Medium
DPoS	Public PL.	Medium	High	High	Medium	<51% validators	Medium	N/A	High	Medium
PoI	Public PL.	High	High	High	Medium	<51% importance	Low	Low	High	Medium
PBFT	Private P.	Medium	Low	High	Low	<33% Faulty Replicas	Low	High	High	High
dPBFT	Private P.	Medium	High	High	Medium	<33% Faulty Replicas	Low	High	High	Medium
Stellar	Public PL.	High	High	High	Medium	Variable	Low	Medium	High	Medium
Ripple	Public PL.	High	High	High	Medium	<20% Faulty UNL Nodes	Low	Medium	High	Medium
Tendermint	Private P.	Medium	High	High	Low	<33% voting power	Low	High	High	Medium
OmniLedger	Public PL.	High	High	High	Medium	<25% faulty Validators	Medium	Medium	Low	Medium
RapidChain	Public PL.	High	High	High	Medium	<33% faulty validators	Medium	Low	Low	Medium
Raft	Private P.	Medium	High	High	Low	<50% crash fault	Low	N/A	High	Medium
Tangle	Public PL.	Medium	High	High	Low	<33% computing power	Low	Low	Low	High

Tabella 2-prestazioni degli algoritmi di consenso

- *Accessibility:* Nelle supply chain spesso è necessario che tutti i partecipanti siano autorizzati prima di poter partecipare nella rete, per cui si utilizzano principalmente le blockchain permissioned. Bisogna però ricordare che le permissioned hanno bisogno di servizi di verifica dell'identità / autorizzazioni, noti come notai (che possono essere sia centralizzati che decentralizzati).

- *Decentralization:* avere un qualunque tipo di controllo su chi può o meno far parte della blockchain può limitare il concetto di decentralizzazione tipico dell'idea originale di Bitcoin. Detto ciò, la necessità di una maggiore privacy interna nelle aziende, fa sì questo fattore assuma minor importanza nell'ambito di una Supply Chain (per cui non deve per forza essere High).

- *Scalability:* l'importanza di questo fattore cresce al crescere del numero di nodi che prendono parte alla blockchain. Tale numero cambia sia a seconda del numero di dispositivi IoT che interagiscono direttamente con la blockchain, che a seconda del numero di attori che partecipa alla supply chain. Inoltre, bisogna valutare anche la possibilità che vi siano espansioni del numero di questi ultimi.

- *Throughput:* sarebbe il numero di transazioni che possono essere processate dall'algoritmo nell'unità di tempo. I valori sono assegnati come segue:

    1. **'low'**: meno di 100 TPS (transazioni per secondo).
    2. **'medium'**: TPS tra 100 e 1000.
    3. **'high'**: più di 1000 TPS.

  Il valore necessario in questo campo varia a seconda del tipo e della quantità di dati che deve essere trasmesso dai sensori IoT alla blockchain e quindi della frequenza con la quale vengono trasmessi. Generalmente deve avere un valore alto.

- *Latency:* è il tempo che viene impiegato mediamente dai nodi per raggiungere il consenso. In questo caso il valore è definito come:

1. **'low'**: ordine dei millisecondi.
2. **'medium'**: ordine dei secondi.
3. **'high'**: ordine dei minuti.

Le reti IoT in genere consistono di molti dispositivi che devono comunicare tra loro, per cui la latenza è importante che non sia bassa (anch'essa dipende comunque dal tipo di dato).

- *Adversary tolerance:* l'importanza di questo fattore è massima nelle blockchain pubbliche, dove il rischio che vi siano nodi malintenzionati è alto. Questo è però molto alleviato nel caso di una rete privata, dove i membri sono conosciuti.
- *Computing overhead:* nel caso in cui i blocchi debbano essere approvati dai dispositivi IoT stessi, arrivando al consenso, la potenza computazionale ammessa sarà ridotta. Infatti, ad esempio la classica Proof-of-Work richiede cicli di CPU fuori portata. Bisogna dunque optare per soluzioni che fanno meno riferimento a algoritmi meno 'aggressivi' da questo punto di vista.
- *Network overhead:* Molti algoritmi (es. PBFT) per raggiungere il consenso presentano la necessità di eseguire una fitta rete di comunicazioni tra nodi. Questo riduce la capacità di questi di mantenere bassa la latenza. Ne va dunque privilegiato un livello basso.
- *Storage overhead:* Così come la capacità computazionale, nei dispositivi IoT è presente anche una limitata capacità di immagazzinamento. Bisogna dunque optare per soluzioni che permettano di limitarne la richiesta. Ad esempio, la Proof-of-Capacity, dove la priorità per il mining dei blocchi si

- ottiene con la regola della 'maggior capacità hard disk' (al posto del riferimento di potenza computazionale della PoW), non è adatta per l'applicazione con IoT.

		DLT					
		Hyperledger Fabric	**Hyperledger Sawtooth**	**Bitcoin**	**Ethereum**	**Corda**	**Iota**
FEATURES	**Consensus method**	Pluggable (PBFT generally)	Proof of Elapsed Time	Proof of Work	Ethash (PoW), Casper (PoS)	Pluggable (raft generally)	Distributed PoW
	Accessibility	Private	Private	Public	Public	Private	Public
	Mode of operation	Permissioned	Permissionless & Permissioned	Permissionless	Permissionless & Permissioned	Permissioned	Permissionless
	Decentralization	Partially	Partially	Yes	Yes	Partially	Partially
	Compute-intensive	No	No	Yes	Partially	No	No
	Network-intensive	Yes	No	No	No	No	No
	Scalability	Low	High	High	High	Partially	High
	Throughput	High	High	very low	low	High	High
	Latency	100 ms	Very Low	10 Minutes	12 seconds	Very Low Not measured	10 ms
	Immutability	Low	Low	High	High	High	High
	Adversary Tolerance	33% Faulty Replicas	unverified	<25% computing power	<51% stakes	unverified	33% computing power
	Privacy	High	High	Low	Low	High	Low
	Smart contract	Yes	Yes	Limited	Yes	Yes	No
	Currency	None but tokens possible	None but tokens possible	Bitcoin (BTC)	Ether (ETH), tokens possible	None	Iota

Tabella 3-principali caratteristiche di alcuni framework DLT

Dunque, dopo aver verificato la necessità di una soluzione che utilizzi queste tecnologie e prima di iniziare lo sviluppo della PoC, è importante valutare il livello che i parametri sopra riportati devono necessariamente possedere. Tenendo a mente queste indicazioni, si può eseguire un'analisi dei framework di DLT esistenti tenendo conto delle necessità richieste dal caso in esame. Così facendo è più semplice capire quali soluzioni scartare e quali considerare nella scelta.

Da qui notiamo come sia anche importante conoscere che funzioni si vogliono implementare per regolare i rapporti tra i vari attori della supply chain. Ad esempio, normalmente per poter tracciare un prodotto lungo tutta la sua vita (dal fornitore della materia prima al cliente) all'interno di una blockchain, si deve creare un suo 'corrispondente digitale'. Per fare ciò di solito si usa un sistema chiamato 'tokenization': significa dunque rappresentare un qualsiasi asset come un token che può essere salvato all'interno della blockchain. A questo vengono assegnati metadati come provenienza, certificazioni (di qualità o altre), storia del processo di trasporto del prodotto stesso. Quando il prodotto si muove da un attore all'altro, il passaggio di proprietà viene rilevato da un sensore IoT e il suo token viene trasferito (esattamente come fosse una transazione).

Altro aspetto utile è infatti la possibilità di creare smart contracts, algoritmi inseriti all'interno della blockchain che seguono l'andamento di un o più dati (trasmessi dal dispositivo IoT stesso, ad esempio), e sulla base del valore eseguono un'azione predefinita. Come nel caso del 'passaggio di proprietà' di un token, queste funzionalità possono essere di fondamentale importanza nell'implementazione inerente alla supply chain.

Come detto prima, nei rapporti tra aziende è importante che rimanga alto il livello di privacy dei dati (spesso alcuni di questi sono sensibili). Perciò, deve essere possibile permettere o meno l'accesso ad alcune informazioni, a seconda dell'attore che vi accede.

'Filtrando' questi dati a seconda delle necessità della supply chain, è possibile dunque adottare la soluzione blockchain che meglio le soddisfa.

4. IOTA: DLT per l' Internet of Things

Ricapitolando ciò che abbiamo visto finora, si possono evidenziare alcuni step fondamentali per la valutazione e scelta del corretto framework DLT da adottare in un caso specifico:

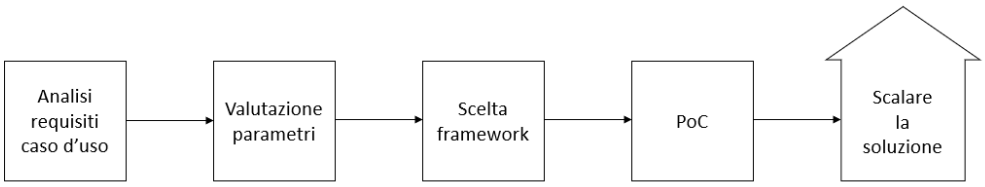


Figura 9-processo di selezione e implementazione di una soluzione con DLT

Dopo aver preso in considerazione e analizzato nel dettaglio questi fattori e le necessità relative ad un'applicazione per il track and trance di una supply chain, la soluzione adottata per lo sviluppo di questa PoC è IOTA: un DLT aperto, feeless e scalabile, progettato per supportare il trasferimento di dati e valore in maniera rapida ed efficiente. Bisogna sottolineare che in questa soluzione le transazioni non sono strutturate a blocchi, ma sono collegate singolarmente tra loro a formare non una catena ma bensì una rete, detta Tangle. IOTA dunque non è una blockchain.

Dalla tabella presentata nel capitolo precedente si nota che il framework presenta ottime prestazioni per quel che riguarda quantità di computazione richiesta, scalabilità, throughput, latenza, immutabilità e tolleranza ad attacchi.

La scelta di una soluzione permissionless, nonostante vada contro la preferenza per i framework permissioned espressa dalle aziende, è stata dettata dal fatto che IOTA mette a disposizione una sovrastruttura di nome MAM che permette all'utente di pubblicare le transazioni all'interno di un canale criptato e quindi di fatto privato. Inoltre, la struttura aperta del Tangle è proprio la caratteristica che permette di registrare prestazioni così alte.

Utilizzando IOTA, è possibile costruire applicazioni che beneficiano di quanto segue:

- *Autenticità*: ogni nodo può dimostrare di aver inviato dati e/o di possedere token IOTA. Ogni nodo di una rete IOTA convalida le transazioni, per poi inviarle ad altri nodi che fanno lo stesso. Di conseguenza, le transazioni valide vengono concordate da tutti i nodi, eliminando la necessità di doversi affidare a un solo nodo nella rete.

- *Integrità*: ogni nodo può dimostrare che i vostri dati non sono cambiati. Tutte le transazioni nel Tangle sono immutabili e trasparenti. Ogni transazione fa riferimento agli hash di altre due che la precedono. Quindi, se il contenuto di qualsiasi transazione dovesse cambiare, gli hash non sarebbero validi, rendendo le transazioni non valide.

- *Riservatezza*: ogni utente può controllare chi ha accesso ai dati attraverso la crittografia. IOTA utilizza firme a utilizzo singolo per impedire agli aggressori di rubare i token IOTA. Le reti IOTA sono reti peer-to-peer in cui nessuna autorità centrale controlla il Tangle. Al contrario, tutti i nodi possono possederne una copia e raggiungono un consenso sul suo contenuto.

- *Micropagamenti*: Inviare piccoli importi di token IOTA senza pagare alcuna fee. IOTA è infatti completamente gratuito. Non è necessario pagare un abbonamento, né firmare alcun contratto. Anche le transazioni sono feeless. È possibile memorizzare i dati nel Tangle senza restrizioni. Tutto ciò che serve è un nodo a cui inviare le transazioni. Per ogni transazione allegata al Tangle, vengono convalidate due transazioni precedenti. Questo processo rende la struttura incredibilmente scalabile perché un numero maggiore di nuove transazioni porta a convalide più rapide e ad una maggiore sicurezza della rete.

Sebbene nel contesto di una supply chain sia utile e importante poter disporre di Smart Contracts, che allo stato attuale IOTA non possiede, per il caso che è stato

esaminato quest'aspetto non è stato considerato come rilevante e quindi non valutato nella scelta del framework. Tuttavia, la fondazione IOTA sta lavorando allo sviluppo di una forma di Smart Contracts che verrà rilasciata nei prossimi mesi, e che sarà trattata nel paragrafo 4.4: Futuri Sviluppi (pag.90). Riassumendo, anche se le blockchain e il Tangle rientrano entrambi nella categoria dei DLT, ma con due principali differenze, che sono:

- Il Tangle non ha costi per le transazioni transazione
- Le reti IOTA non hanno miners

Per spiegare questi punti, è necessario comprendere le differenze tra le strutture dei dati e i meccanismi di consenso di questi DLT.

4.1. Componenti del Tangle

4.1.1. La rete delle Transazioni di IOTA

Il Tangle è la struttura immutabile che contiene una cronologia aggiornata di tutte le transazioni eseguite su IOTA. Tutti i nodi della rete memorizzano una copia del Tangle e raggiungono il consenso riguardo al suo contenuto.

Per rendere tale struttura immutabile, ogni transazione in esso contenuta è legata a due precedenti transazioni per mezzo dei relativi hash.

Questi collegamenti formano una struttura di dati chiamata Directed Acyclic Graph (DAG) simile a quella sotto rappresentata:

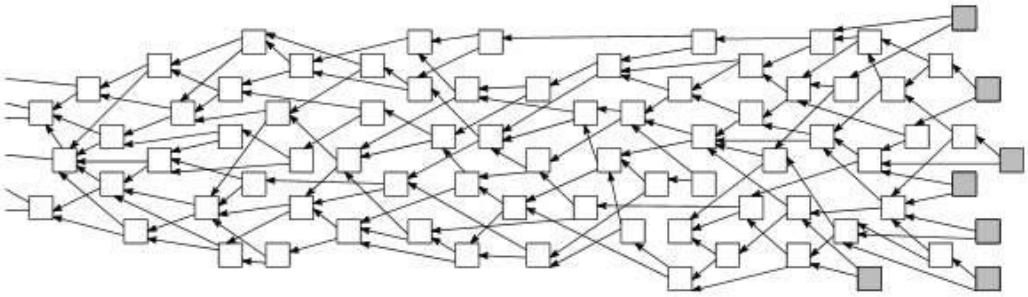


Figura 10-struttura del Tangle di Iota

Qui le transazioni a sinistra vengono cronologicamente prima delle transazioni a destra.

All'interno del DAG, sono chiamate 'tips' le transazioni non ancora confermate presenti all'interno del tangle. La prima transazione, alla quale tutte le altre sono collegate, è invece chiamata transazione di genesi.

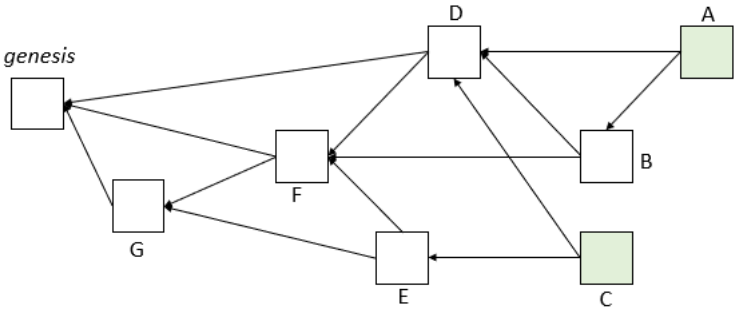


Figura 11-tips del Tangle

L'altezza (Height) è la lunghezza del percorso più lungo fino alla transazione di genesi. Per esempio, nella seguente immagine l'altezza della transazione G è 1, mentre D ha un'altezza pari a 3.

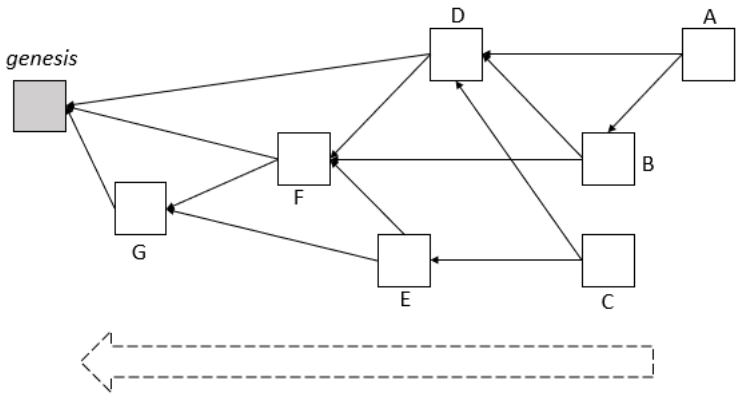


Figura 12-altezza dal tip

La profondità (Depth) è invece il numero di transazione che intercorrono nel più lungo percorso inverso fino ad un tip. Nel caso sottostante dunque, G ha una profondità di 4 transazioni fino al tip A, che è il più 'lontano'.

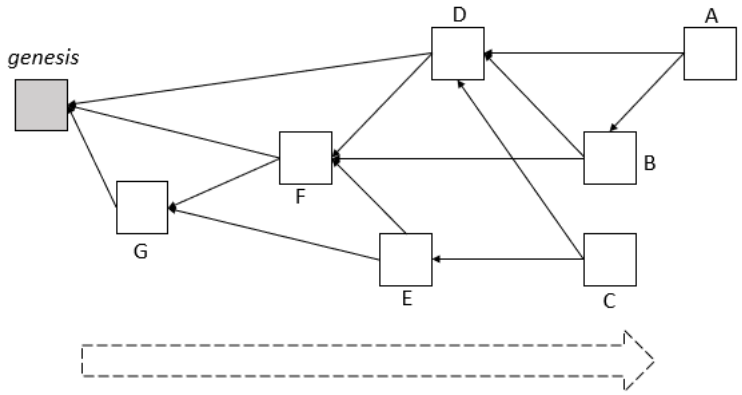


Figura 13-profondità dalla genesi

4.1.2. La rete dei Nodi di IOTA

I **nodi** rappresentano le componenti base che formano la struttura di una rete IOTA. Essi eseguono il software che dà loro accesso alla lettura e scrittura sul Tangle (IRI). Come ogni sistema distribuito, ogni nodo è collegato ad altri chiamati vicini

(*neighbors*) per formare una rete. Quando un nodo, indipendentemente da dove si trova nel mondo, registra una nuova transazione, cerca di inoltrarla a tutti i suoi vicini. In questo modo, tutti i nodi alla fine possono convalidare le transazioni e memorizzarle, dopo aver controllato se essa sia presente o meno nella propria versione del Tangle. Infatti, poiché i nodi sono distribuiti in tutto il mondo, potrebbero avere transazioni diverse nei loro ledger in qualsiasi momento. Per assicurarsi ciò non accada, questi devono sincronizzarsi con il resto della rete. Un nodo si dice **sincronizzato** quando ha *solidificato* tutti i **milestone** fino all'ultimo.

Nel software dei nodi che eseguono la Mainnet, un nodo si considera sincronizzato quando i valori di 'latestMilestoneIndex' e 'latestSolidSubtangleMilestoneIndex' sono uguali.

La '**solidificazione'** è il processo in cui un nodo chiede ai suoi vicini la storia di tutte le milestone del Tangle, partendo da una 'entry point milestone' e terminando con l'ultima. Quando un nodo ha la storia di tutte le milestone fino alla prima, la segna come solida e ricomincia il processo da quella successiva. Di conseguenza, più vecchia è la milestone dell'entry point, più tempo richiederà la solidificazione.

Grazie al fatto che nodi hanno la possibilità di creare versioni locali del Tangle, è possibile per loro staccarsi e rientrare nella rete. Questo processo si chiama *partizionamento*.

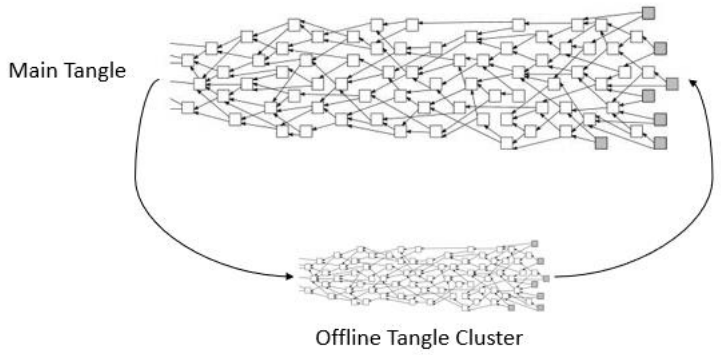


Figura 14-porzione offline di Tangle

Per esempio, se dei nodi connessi a sensori sui container delle navi da carico dovessero perdere la connettività con il Tangle principale quando la nave attraversa l'oceano, possono crearne una versione offline e poi collegarlo una volta tornata la connessione.

4.1.3. Velocità del Consenso

Nel Tangle, poiché la velocità di validazione della rete è strettamente legata alla quantità di transazioni emesse, la conferma di queste può essere raggiunta in una frazione del tempo necessario per le Blockchain. Più attività c'è nel Tangle, più convalide si verificano e più veloce ne diventa la finalizzazione. Infatti, nella Blockchain viene sempre approvato un blocco alla volta, e questo sarà confermato solo dopo che avrà circa 6 figli; la velocità di approvazione dei blocchi dipende dunque solo dalla potenza computazionale dei minatori (i blocchi verranno confermati più velocemente solo all'aumentare della potenza). In Iota invece posso approvare più transazioni contemporaneamente. Dato che una transazione è considerata completa solo dopo che ha un certo numero di figli, più transazioni avvengono, più aumenterà la potenza di calcolo cumulata e più velocemente verrà raggiunta l'approvazione, perché più velocemente si creerà una catena di figli sufficientemente grande collegata ad ogni transazione.

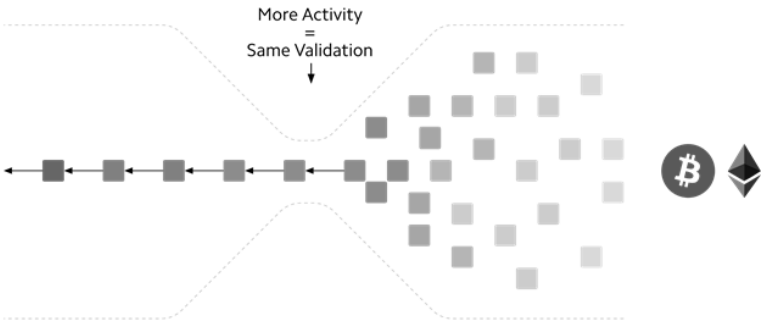


Figura 15-il collo di bottiglia della blockchian, causato dall'approvazione del blocco

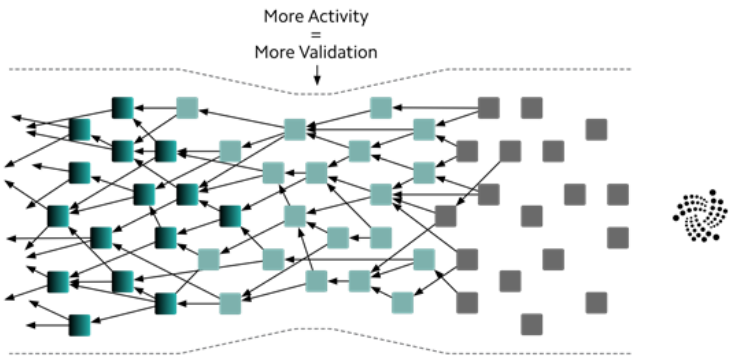


Figura 16-Iota è maggiormente scalabile rispetto ad una blockchain

4.1.4. Sicurezza

La sicurezza è una caratteristica intrinseca di tutte le tecnologie decentralizzate, distribuite e senza permessi: la capacità di un aggressore di esercitare un'influenza maligna sulla rete è bassa perché le risorse necessarie per essere in grado di farlo sono troppo grandi.

In una Blockchain però è protetta dalla PoW cumulata dei soli miners. IOTA invece funziona in modo fondamentalmente diverso. Ogni transazione richiede una minima quantità di PoW e/o richiede una minima larghezza di banda; è dunque il traffico stesso della rete a proteggerla da attacchi. Più transazioni si verificano, più la rete diventa sicura, perché maggiore sarà la quantità di potenza computazionale che un nodo maligno dovrà possedere singolarmente per sopraffare quelli onesti.

Attualmente, tuttavia, il basso throughput delle transazioni rende falsa quest'ipotesi. Le misure di sicurezza aggiuntive, come le milestones e il Coordinator sono infatti necessarie per rendere sicuro il Tangle nei suoi primi tempi, come indicato all'interno del White Paper scritto da Serguei Popov.

4.1.5. Snapshot

IOTA funziona su una rete pubblica. Per 24 ore al giorno, 7 giorni alla settimana, chiunque può memorizzare gratuitamente quantità illimitate di dati sul Tangle (con solo una piccola PoW per transazione)

Di conseguenza, le dimensioni del Tangle sono sempre in crescita, e per mantenerne lo spazio di archiviazione sono necessari costi molto alti. Per evitare che questi diventino troppo grandi, i nodi spesso eseguono dei *local snapshot*.

Un local snapshot è il processo in cui un nodo registra lo stato del suo libro mastro in file locale. Grazie a ciò, il processo di sincronizzazione con i vicini è molto più veloce, perché di fatto il Tangle contiene meno transazioni. Infatti, si raggruppano così diversi trasferimenti allo stesso indirizzo in 1 record unico, si salvano le transazioni a valore positivo e si rimuove la cronologia delle restanti transazioni. Rimane di fatto un'istantanea di ogni indirizzo con il corrispondente saldo non-zero.

Per molti casi di utilizzo aziendale, i dati del tangle IOTA devono però essere conservati per lunghi periodi di tempo. Ad esempio, i dati finanziari devono essere conservati per 10 anni in alcuni casi, e i dati identificativi devono essere conservati per tutta la vita dell'identità. Conservare tali informazioni è possibile, installando un *permanode*, ovvero un nodo che anziché fare snapshot locali, memorizza la storia completa del Tangle.

4.1.6. Wallet

Un portafoglio IOTA è un portafoglio deterministico, nel senso che quando viene generato un nuovo indirizzo viene calcolato dalla combinazione del *seed* [7]e dell'*indice* attuale dell'indirizzo, che può essere un qualsiasi numero intero positivo. Ogni volta che deve essere eseguita una transazione, il wallet partendo dall'indice

[7] Password univoca appartenente ad un wallet, usata per generare indirizzi, chiavi private e firme delle transazioni. Deve essere perciò utilizzato con cura dal possessore, che deve curarsi di non condividerlo con nessuno.

“0” crea indirizzi nuovi e chiede al nodo a cui è collegato se esistono transazioni che li hanno utilizzati. Se non si trovano transazioni che vi fanno riferimento, si conclude che questo non è ancora stato utilizzato. Se invece vengono trovate transazioni che vi fanno riferimento, il portafoglio aumenterà l'indice di 1 unità è rifarà lo stesso controllo sul nuovo indirizzo. Una volta pubblicata una transazione ad un determinato indirizzo,

- il nodo sceglie e convalida due transazioni del Tangle con una piccola PoW

- nel wallet viene aggiunto l’indirizzo ad una cronologia

4.1.7. Gli Indirizzi

Un *indirizzo* è come un conto bancario che appartiene a un seed e al quale corrisponde un saldo di coins IOTA pari o superiore a 0. Gli indirizzi sono infatti la metà pubblica di una coppia di chiavi pubbliche/private, similmente a come funziona Bitcoin. Per trasferire i token IOTA da un indirizzo a un altro, si firma una transazione con la chiave privata, dimostrandone la proprietà agli altri nodi. In questo modo potete condividere gli indirizzi con chiunque, perché solo il proprietario del seed conosce la chiave privata.

Ogni indirizzo è caratterizzato anche un livello di sicurezza, oltre che dal suo indice:

- *Indice*: Numero compreso tra 0 e 9.007.199.254.740.991

- *Livello di sicurezza*: Numero tra 1 e 3

Lo stesso seed, lo stesso indice e lo stesso livello di sicurezza, daranno come risultato sempre allo stesso indirizzo. A livello logico, possiamo dunque dire che:

$$indirizzo = seed + indice + livello\ di\ sicurezza$$

Allo stesso modo, lo stesso seed con un indice e/o un livello di sicurezza diverso darà luogo a un indirizzo diverso.

Si consiglia di utilizzare gli indirizzi una volta sola. Infatti, il riutilizzo di uno di essi, soprattutto per le transazioni in uscita, può avere enormi implicazioni sulla sicurezza del proprio wallet.

Per questo motivo, nel momento in cui faccio una transazione di coin dall'indirizzo che usa l'indice 'x', IOTA crea automaticamente un nuovo indirizzo, utilizzando l'indice successivo a quello corrente, e si sposta il denaro contenuto all'indice x in quest'ultimo. Ciò significa che non c'è limite al numero di transazioni che un indirizzo può ricevere, ma non appena si sono utilizzati i fondi di quell'indirizzo per effettuarne una, questo non dev'essere utilizzato.

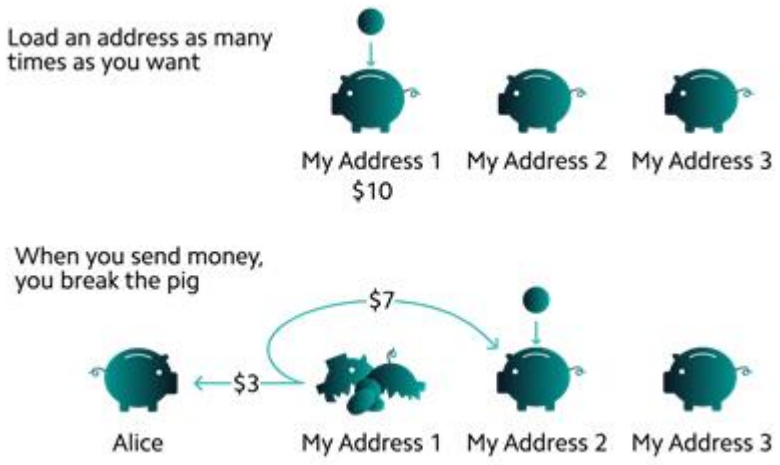


Figura 17-motivo per cui bisognerebbe evitare di riutilizzare un indirizzo già utilizzato per inviare denaro

Infatti, la sicurezza di una transazione diminuisce quando si inviano tokens più di una volta dallo stesso indirizzo. IOTA infatti utilizza le firme ad uso singolo di Winternitz, e in quel frangente viene rivelata una parte (50%) della chiave privata di quell'indirizzo. Più transazioni in uscita si effettuano dallo stesso indirizzo, più facile sarà per gli aggressori rubarne il contenuto, risalendo alla chiave privata. Quindi, nel caso di un oggetto che si muove tra diversi attori di una supply chain, dovrò creare un nuovo indirizzo ogni volta che aggiorno la posizione di quest'ultimo inviando dati al tangle da un nodo.

Per “attivare” un indirizzo e mostrare che questo è attivo, normalmente si esegue una transazione di valore zero ad esso, facendo si che sia dunque visisbile sul Tangle.

4.1.8. Struttura di una Transazione

Ogni transazione che viene pubblicata sul Tangle è contrassegnata da un hash che la identifica. Di fatto, una transazione è una singola istruzione che può depositare o prelevare dei token IOTA da un indirizzo, oppure avere valore zero e contenere dati, un messaggio o una firma. Questo argomento descrive i tipi di transazione, le differenze tra loro e la struttura di una transazione.

Di fatto le strutture che vengono collegate al Tangle sono però dei “pacchetti”, detto *bundle*, di più transazioni tra loro indivisibili, ovvero che possono solo essere o accettate o rifiutate tutte quante. Ad esempio, una transazione che deposita token IOTA in un indirizzo sarà legata ad un'altra transazione che si occupa di ritirare quei token da un altro indirizzo. Pertanto, queste transazioni devono essere nello stesso bundle.

Prendiamo ad esempio il pacchetto di una transazione che viene pubblicata sul Tangle. Per fare ciò dovrà confermare due tip, che chiameremo *tip 0* e *tip 1*. Questa transazione sarà caratterizzata da diversi campi, che ne definiscono le caratteristiche:

- *Hash*: serve a identificare in modo univoco la transazione sul Tangle. Questo valore viene generato svolgendo un hashing della transazione in trits.
- *signatureMessageFragment*: Contiene una firma o un messaggio, il contenuto dei quali può essere anche frammentato su più transazioni in un bundle. Se la transazione registra una spesa di un certo valore, allora ne viene eseguita obbligatoriamente anche la firma digitale. Se invece la transazione contiene solo un messaggio, non è richiesta invece alcuna firma.
- *Address*: L'indirizzo associato a questa transazione.

- *Value*: Se il campo ha un valore >0, allora il campo "*address*" indica l'indirizzo del destinatario. Altrimenti, conterrà l'indirizzo del mittente.
- *Timestamp*: timestamp di quando la transazione è stata pubblicata.
- *currentIndex*: indica la posizione che la transazione ha tra quelle all'interno dello stesso *bundle.* Se questo valore è uguale a zero, la transazione è chiamata "*tail transaction*", se è uguale al valore indicato nel campo *lastIndex,* allora sarà chiamata "*head transaction*".
- *lastIndex*: l'ultima posizione che la transazione può assumere nel bundle. Il numero totale di transazioni contenute sarà dato da:

$$lastIndex + 1$$

- *bundle*: l'hash del bundle, che indica quali transazioni appartengono allo stesso bundle. Il valore è generato facendo l'hash dei metadata relativi a tutte le transazioni del bundle.
- *trunkTransaction*: se la transazione corrente è la *head* transaction, allora questo campo indica il valore del *tip 0*, altrimenti indica il valore dell'hash della transazione seguente del bundle.
- *branchTransaction*: se la transazione corrente è la *head* transaction, allora questo campo indica il valore del *tip 1*, altrimenti indica il valore del *tip 0*.
- *tag*: nome definito dall' utente per aiutarlo a trovare la transazione. Questo campo può anche rimanere vuoto.

Inoltre, le transazioni facenti parte di uno stesso bundle possono essere di uno dei seguenti tre tipi:

- *Transazione di input*: contengono l'istruzione di ritirare i token IOTA da un indirizzo. Per essere valida deve sempre contenere: un valore negativo nel campo "*value*"; un indirizzo (inidicato al campo *address*) che contenga almeno l'importo indicato nel campo "*value*"; almeno il primo frammento di una firma valida nel campo "*signatureMessageFragment".*
- *Transazione di output*: le transazioni in cui i token sono invece inviati a uno o più indirizzi. Queste transazioni sono facilmente riconoscibili perché il

valore della transazione è sempre superiore a 0 e l'indirizzo non appartiene al mittente.

- *Transazione a valore zero*: come anticipato, all'interno del *signatureMessageFragmet* possono contenere in maniera indifferente un frammento della firma digitale oppure del messaggio. L'identificazione non è obbligatoria, visto che il contenuto della transazione non prevede uno scambio di Token.

4.2. Funzionamento delle transazioni nel tangle

Effettuare una transazione è un processo composto da 3 fasi:

- ***Firma***: Il vostro nodo (computer / cellulare) crea una transazione e la firma con la chiave privata.

- ***Selezione del tip***: Il vostro nodo sceglie altre due transazioni non confermate (tips) secondo ciò che risulta utilizzando l'algoritmo Weighted Random Walk (WRW).

- ***Approvazione delle transazioni*** *tramite Proof-of-Work*: il nodo controlla che le due transazioni non siano in conflitto, ovvero che non siano verificati casi di double spending tra di esse. Successivamente, il nodo deve eseguire una piccola quantità di PoW risolvendo un puzzle crittografico (*hashcash*), che serve a prevenire gli attacchi di spam[8] e di Sybil[9]. Lo scopo primario è quello di imporre un costo arbitrario, anche se piccolo, per l'emissione di una transazione, costringendo il nodo che pubblica a trovare un "nonce" con un certo livello di difficoltà. Questo rende troppo dispendioso se non praticamente impossibile per un attore malintenzionato generare una parte sostanziale del totale delle transazioni che avvengono sulla rete

[8] Lo **spam** avviene quando si tenta di sovraccaricare un sistema con l'invio di messaggi ripetuti ad alta frequenza.
[9] Gli attacchi Sybil si verificano quando una rete di computer viene attaccata inserendo nodi falsi o dannosi.

IOTA, in numero tale da poterne superare la potenza computazionale. Questo perché la potenza di calcolo è una risorsa scarsa e costosa; infatti, un aggressore non può accumulare più potenza di calcolo della potenza di calcolo cumulativa dei nodi onesti, perché questi sono la maggioranza.

Il fatto che ogni nodo esegua una piccola PoW ad ogni transazione eliminal la necessità di ricompensare miners; in IOTA ogni nodo è un miner e il suo interesse sta proprio nel fatto che più transazioni avvengono, più la rete è sicura e più rapida.

4.2.1. Selezione del Tip

L'obiettivo dell'algoritmo ***Weighted Random Walk (WRW)*** è quello di generare campioni equi a partire da una distribuzione difficile e all'interno di IOTA è utilizzato in due modi:

- per selezionare altre due transazioni non ancora confermate (suggerimenti) quando si crea una transazione, in maniera tale da verificarne la veridicità.

- per determinare se una transazione è confermata.

Per scegliere quali transazioni devono essere approvate da un nodo, un '*walker'* percorre a ritroso il Tangle dalla transazione di genesi fino ad arrivare ad un tip, la quale verrà scelta e su di essa verrà eseguita la verifica. La scelta della strada che deve essere percorsa da quest'ultimo è pesata, non completamente random. Infatti, la strada che verrà scelta sarà sempre quella più lunga: il *walker* si muoverà sulla transazione che ha un *peso cumulato* maggiore con maggiore probabilità, dove il peso è dato dal numero di transazioni che approvano una data transazione x.

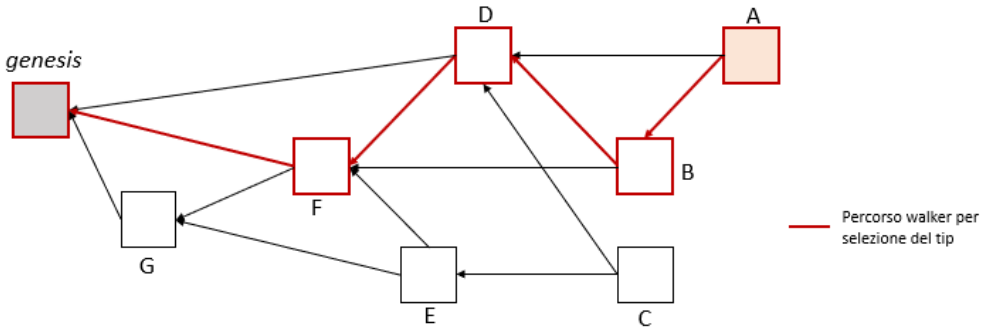


Figura 18-percorso pesato del walker dalla genesi al tip

Nel seguente esempio, la transazione 6 è stata scelta rispetto alla transazione 8 e alla 7 nel percorso a ritroso. Questo in quanto è stata approvata da un numero cumulato (quindi direttamente o indirettamente) maggiore di transazioni rispetto agli altri due casi. Dunque, i tip ad essa collegati avranno maggiore possibilità di essere approvati da una nuova transazione (in questo caso, non la 10).

Questo metodo diminuisce le probabilità una nuova transazione (es. 14) scelga di confermare transazioni più vecchie (e quindi di non mantenersi aggiornata rispetto alla situazione del tangle). Questo perché **essa stessa avrebbe poche probailità di essere selezionata,** successivamente. Se ad esempio si provasse a confermare la transazione 2, il walker muovendosi a ritroso e una volta arrivato sopra quest'ultima, dovrebbe decidere se muoversi su una transazione con peso 1 (la ipotetica transazione n°14) oppure su altre transazioni (es. 3, 4, 5) con peso molto maggiore. Le probabilità che si muova su 14, e che di conseguenza questo venga scelto per essere confermato sono molto minori.

Ovviamente, non devono essere scelte solo le transazioni con peso maggiore, altrimenti alcune transazioni rimarrebbero per sempre allo stato di tips, senza mai venire approvate. Di seguito vediamo un caso di questo tipo:

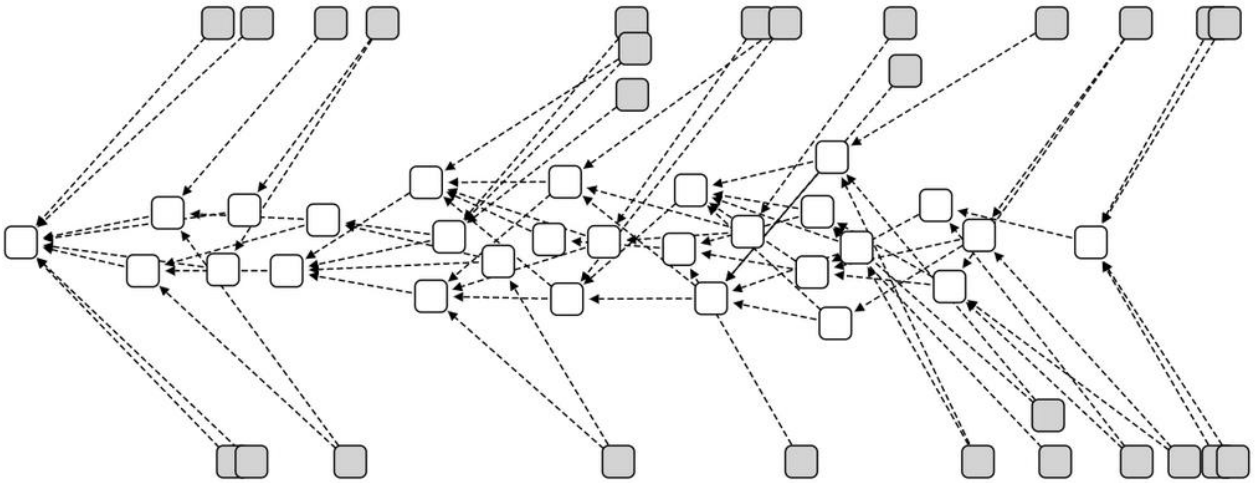


Figura 19-caso in cui il walker si sposti solo su transazioni di peso massimo

Nella scelta del percorso viene dunque inserita una variabile α, che indica quanto influente sulla scelta sia il peso. Se il suo valore fosse 0 allora il peso non avrebbe nessuna importanza, e dunque ogni transazione avrebbe la stessa possibilità di essere scelta, se invece fosse alto, il peso avrebbe allora troppa importanza e dunque si rischia di rientrare nella situazione sopra citata. Il giusto bilanciamento di questo valore è definito dalla regola *Markov Chain Monte Carlo*.

4.2.2. Approvazione delle Transazioni

Ipotizziamo una situazione in cui un utente A (Alice) vuole effettuare una transazione in IOTA ad un utente B (Bob). Questo generalmente non può portare il primo ad avere un account con balance negativo, e bisogna dunque verificato che ciò non avvenga. Inoltre, dovrò verificare anche tutte le transazioni in precedenza approvate da Alice, per certificare che nessuna di queste sia negativa (fino a quella di genesi).

Per ipotesi, cosa accadrebbe se Alice tentasse di eseguire una transazione di valore superiore a quello che effettivamente possiede? Come mostrato di seguito, Bob non potrebbe approvare entrambe le transazioni di Alice, perché esse comportano un

saldo negativo. Farlo comporterebbe un'infrazione delle regole del protocollo IOTA, e gli altri utenti della rete non approveranno di conseguenza la sua transazione (è chiaro che questo problema si pone solo quando le transazioni hanno un valore monetario).

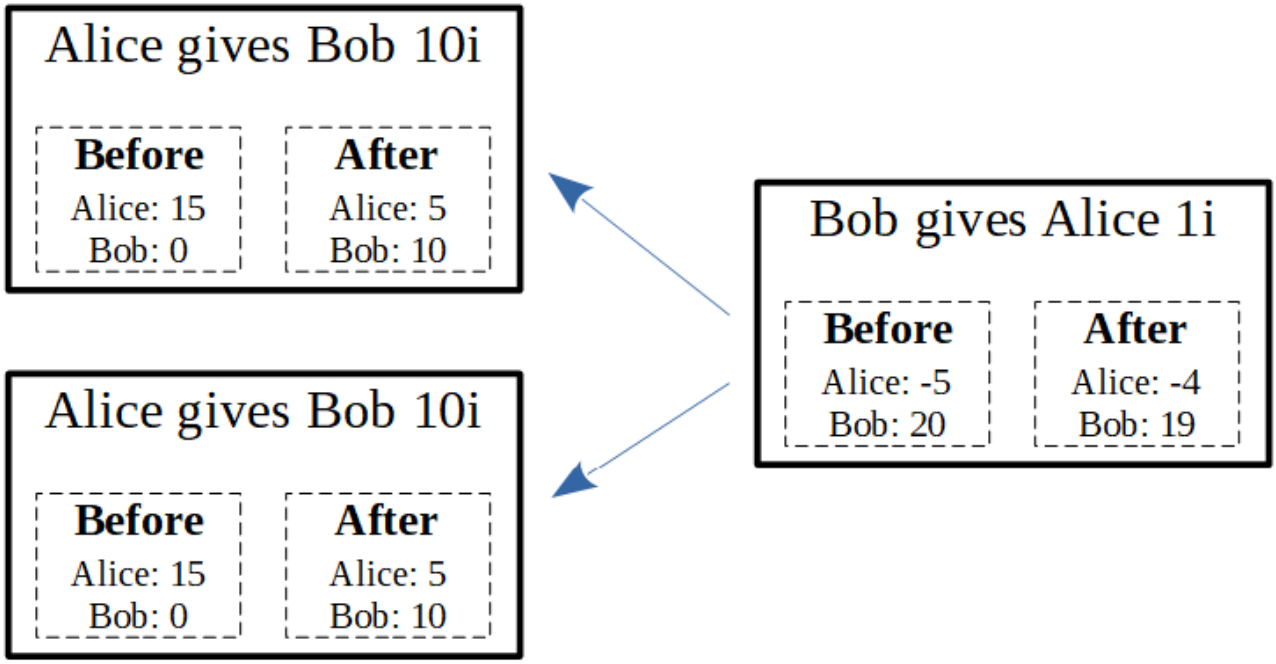


Figura 20-doppia spesa su Iota

Questo caso è conosciuto col nome di "doppia spesa", perché Alice ha di fatto speso i suoi soldi due volte. Forse non intendeva nemmeno farlo, ma ha inviato la sua transazione due volte per errore prima che questa venisse approvata e i suoi soldi le venissero di fatto scalati. Nonostante ciò, ha creato due rami nel tangle che non possono essere riconciliati, e ci pone davanti una domanda: quale ramo dovrebbe essere approvato dagli utenti IOTA?

La soluzione a questo problema è ancora una volta la 'Weighted Walk' di cui abbiamo parlato prima. La scelta sarà dunque condizionata dal peso delle singole transazioni. Alla fine, **uno dei due rami diventerà più pesante dell'altro** e quello secondario verrà "abbandonato", proprio perché non sarà più conveniente per chi

pubblica nuove transazioni (vedi paragrafo 4.2.1 pag.70). Ed è proprio per questo che una transazione non può essere considerata confermata subito dopo la sua emissione, anche se ha un certo num4ero di conferme, perché potrebbe essere parte di un ramo del Tangle che alla fine verrà abbandonato. Per essere sicuri che la transazione sia confermata, bisogna aspettare che la sua fiducia sia sufficientemente alta.

Dal punto di vista di chi deve ricevere il pagamento, c'è però un problema: Bob come fa a sapere se ha davvero ricevuto i soldi da Alice?

Immaginate che i pagamenti non siano stati effettuati, come nell'esempio di prima, alla stessa persona, ma a due persone diverse, che questi siano commercianti e che Alice abbia acquistato un prodotto da ognuno di loro. Se entrambi spedissero l'ordine subito dopo aver visto la sua transazione nel Tangle, alla fine uno di loro scoprirebbe di non essere stato pagato. Come possono sapere quando è sicuro eseguire la spedizione?

Questo è un problema, e Bitcoin è stata la prima tecnologia a risolverlo nel 2009. IOTA lo affronta grazie ad un concetto chiamato "fiducia della conferma". Questa fiducia è la misura del livello di accettazione di una transazione da parte del resto del Tangle.

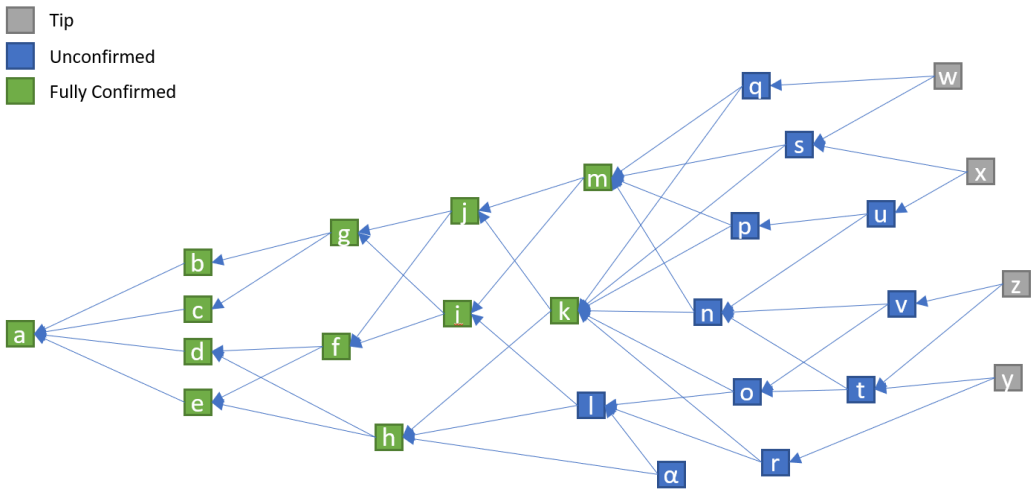


Figura 21-transazioni confermate e non confermate

Nell'immagine qui sopra, si può notare che i blocchi verdi (che sono quelli considerati confermati) sono indirettamente collegati a tutti i blocchi grigi. Perchè

una transazione sia considerata confermata, deve infatti esistere un percorso che la raggiunge a partire da ogni del Tangle tip.

Per determinare lo stato di conferma della vostra transazione è necessario fornire a Iota la profondità da cui partire per eseguire esegue l'algoritmo della WRW un numero N di volte. La probabilità che la vostra transazione venga accettata sarà $\frac{M}{N}$. M è invece il numero di volte che l'algoritmo termina su un tip eseguendo un percorso che passa per la transazione in esame. Se si esegue una WRW 100 volte, e 60 tip hanno un percorso per la transazione, allora la transazione è confermata al 60%. In altre parole, la fiducia di una transazione è la percentuale di tip che la approva.

Questa modalità è simile a ciò che avviene in Bitcoin, dove però si tratta di attendere la conferma di almeno 6 blocchi per poterne considerare uno validato.

Per precisione, bisogna dire che di fatto su IOTA questo procedimento viene effettuato (dal nodo IRI [10] che pubblica la transazione) su un "*subtangle*" di IOTA, ovvero sulla porzione di Tangle, che ha come transazione di genesi l'ultima *milestone* alla quale il nodo in questione è sintonizzato. Quest'ultima è una transazione emessa dalla fondazione IOTA per cui tutte le transazioni precedenti ad essa collegate sono considerate approvate (in seguito vedremo nel dettaglio di cosa si tratta).

4.2.3. Peso delle transazioni

Ogni transazione è caratterizzata da un "*peso proprio*" iniziale che è determinato dallo sforzo computazionale compiuto dal nodo che l'ha immessa ne Tangle, per verificare due transazioni. Il *peso cumulativo* di una transazione è invece dato dal peso proprio della transazione più la somma di tutti i pesi di tutte le altre che direttamente o indirettamente la approvano.

---

[10] L'IRI (IOTA reference implementation) è un software Java open-source per il protocollo IOTA. Questo software funziona attualmente sui nodi delle reti IOTA pubbliche, dove gli utenti possono trasferire il token IOTA tra di loro.

Il peso cumulativo è una metrica di fondamentale importanza per le transazioni in fase di approvazione della rete. Ogni nuova transazione aggiunta al Tangle aumenta il peso cumulativo di quelle venute prima. Le transazioni più vecchie crescono dunque di importanza nel tempo. L'uso dei pesi cumulativi evita lo spamming e altri tipi di attacco attacchi, in quanto come già accennato si presume che nessuna entità possa generare una quantità di transazioni abbastanza elevata da surclassare le altre in un breve periodo di tempo.

La *Minimum Weight Magnitude* (MWM) è la difficoltà minima della PoW che deve essere impiegata da un nodo all'interno di una determinata rete IOTA per effettuare una pubblicazione (similmente al ruolo che l'Hashcash ricopre in Bitcoin).

4.2.4. Milestone

Ora è chiaro come e quando un commerciante sia in grado di capire quando è sicuro inviare un prodotto a chi lo compra. Una volta che la transazione di Alice raggiunge una soglia di fiducia molto alta, diciamo il 95%, può considerarsi verificata ed è impossibile venire truffati.

O meglio, è molto poco probabile, piuttosto che impossibile; se chi esegue la transazione è intenzionato a imbrogliare, e ha abbastanza potenza di calcolo per farlo, può provare ad effettuare una doppia spesa.

Per farlo dovrà approvare due vecchie transazioni con la seconda, che non facciano riferimento a quella originale. Dopodiché inizierà quindi ad emettere quante più transazioni possibili, cercando di aumentare il peso cumulativo del ramo sul quale ha approvato tali transazioni. Se ha abbastanza potenza di calcolo, questo può far sì che l'intera rete di IOTA le creda e prosegua sulla via da lei iniziata, raddoppiando con successo le spese.

Questo scenario è un rischio solo se Alice può inviare più transazioni di tutti gli altri utenti messi insieme, o quasi. Pur non essendo un rischio reale in una rete matura e attiva, può rappresentare un vero problema per IOTA, al suo stato odierno. Infatti, il numero di transazioni al secondo sulla rete principale non è ancora abbastanza alto per escludere questo rischio.

Per motivi di sicurezza, IOTA utilizza dunque un meccanismo di consenso diverso temporaneo: il "***coordinator***[11]". Ogni due minuti, la Fondazione emette una transazione **milestone** e tutte le transazioni da essa approvate (quindi precedenti e confermate direttamente o indirettamente da essa) assumono un livello di fiducia del 100%. In questo modo il coordinator stabilisce la direzione generale di crescita del Tangle. Le transazioni precedenti una milestone (dette transazioni del *confirmation cone*), possono essere di due tipi:

- *state-mutating*: Operazioni di valore che modificano i saldi di due o più indirizzi
- *non-state-mutating*: Transazioni a valore zero.

Le non-state-mutating sono sempre considerate confermate se posseggono la "referenza" ad una milestone. Invece, le state-mutating devono essere verificate, in quanto potrebbero celare delle doppie spese.

In questo modo può essere garantito un riparo da frodi simili a quella descritta prima. Tale sistema sarà utilizzato finché la rete IOTA non sarà cresciuta abbastanza da mantenere sicura da attacchi in maniera decentralizzata, momento in cui l'algoritmo di consenso distribuito completo del Tangle descritto in precedenza sarà del tutto attivo.

Nonostante la posizione (criptata e firmata, dunque non visibile al pubblico) del Coordinatore e il suo codice sorgente non siano pubblici, **l'utilizzo del Coordinatore non significa che la rete sia centralizzata**. Infatti, ogni nodo della rete può verificarne l'attività, confermando che quest'ultimo non stia creando valuta dal nulla o approvando doppie spese. Il Coordinatore sarà infatti eliminato non appena la rete raggiungerà un certo numero di transazioni confermate al secondo (*ctps*).

---

[11] È un nodo client che trasmette transazioni identificabili e riconducibili ad esso chiamate *milestones*, che gli altri nodi della rete riconoscono come affidabili e usano per considerare le transazioni confermate.

Tra tutte le milestone se ne possono nominare alcune di rilevante importanza, che sono:

- **latestMilestone**: L'ultima milestone creata dal Coordinatore. Il valore della latestMilestone è l'hash dell'ultima transazione che il nodo ricevuto dal Coordinatore (di conseguenza, potrebbe essere diverso per ogni nodo, temporaneamente). Il **latestMilestoneIndex** è l'indice dell'ultima milestone.
- **latestSolidSubtangleMilestone** quando il Coordinatore crea le milestone, ogni milestone fa riferimento ad un sottogruppo di transazioni approvate dal Coordinatore. Il latestSolidSubtangleMilestone è l'hash dell'ultima Milstone che il nodo considera solida, e viene inserito all' invio di nuove transazioni. Se venisse utilizzata una SolidMilestone non aggiornata, la transazione potrebbe non essere approvata. Dunque, il nodo che fa le transazioni deve assicurarsi di rimanere sempre aggiornato all'ultima milestone. Il **latestSolidSubtangleMilestoneIndex** è l'indice relativo al latestSolidSubtangleMilestone.

Un full node è sincronizzato con il Tangle se sono soddisfatte le seguenti condizioni:

- Il suo *latestMilestoneIndex* è aggiornato all'attuale lastMilestoneIndex convalidato dal Coordinatore,

- Il sup *latesSolidSubtangleMilestoneIndex* è lo stesso del lastMilestoneIndex.

4.2.5. Processo di Generazione di un Indirizzo

Come detto prima, gli indirizzi di IOTA vengono generati in modo deterministico a partire dal seed (composto da 81 *trytes*[12]).

---

[12] A differenza della maggior parte dei costrutti informatici, IOTA è costruito sulla base di un sistema ternario, anziché binario. Questo può essere bilanciato (dove il trit può avere valori -1,0-1) e non bilanciato (dove i valori sono 0,1,2). IOTA si basa sul *Trit* bilanciato, che è il corrispettivo

Come prima cosa, nel processo di creazine di un indirizzo, bisogna tradurre il seed dall' alfabeto in tryte a trit.

- Seed (tryte): C9RQF ... QIAWT (81 tryte)
- Seed (trits): 0,1,0,0,0,0,0,0 ... -1,-1,0,-1,1,-1 (= 81 x 3 = 243 trits)

Successivamente, bisognerà prendere il corrispondente indice di riferimento (dove l'indirizzo 0 avrà l'indice numero 0, l'indirizzo 1 ha l'indice 1, ecc...)

Si passa dunque a creare un "*subseed*", sommando il valore del seed a quello dell'indice (in trits), come nell'esempio sottostante:

$$
\begin{array}{lll}
Seed: & \mathbf{0}, \mathbf{1}, \mathbf{0}, 0,0,0 \ldots -1, -1,0 & + \\
Indice: & \mathbf{1}, \mathbf{0}, \mathbf{0} & = \\
\hline
Subseed: & \mathbf{1}, \mathbf{1}, \mathbf{0}, 0,0,0 \ldots -1, -1,0 &
\end{array}
$$

Inoltre, dobbiamo ricordarci che IOTA offre 3 livelli di sicurezza diversi (1,2,3), il quale determina il numero di volte che viene eseguita un'operazione di Hash (il quale sarà di tipo *Curl* in IOTA) sul *subseed*. Dunque, un diverso livello di sicurezza allo stesso indice si rifletterà nella creazione di indirizzi diversi.

---

del bit, e il *Tryte* che invece significa Trinary Byte, analogo a byte, e consiste di 3 Trit. Per rendere i tryte più leggibili dall'uomo, il team di IOTA ha creato l'alfabeto dei tryte:
9ABCDEFGHIJKLMNOPQRSTUVWXYZ
Tale alfabeto è composto da 26 lettere latine più il numero 9 (27 caratteri totali), dal momento che 1 tryte può avere 3^3 = 27 diverse combinazioni. (traduzione alfabeto negli annex)

4.3. Masked Authenticated Messaging (MAM)

MAM, che sta per "*Masked Authenticated Messaging*", è una delle strutture più utili che si possono utilizzare combinate al protocollo IOTA. La funzionalità principale che ricopre è quella di legare tra loro transazioni eseguite dallo stesso *seed* (riconducibili dunque ad un'unica entità, es. sensore IoT), in maniera tale da poter ricostruire il percorso da esse fatto e lo storico del valore registrato.

4.3.1. Canali MAM

Per fare ciò, MAM utilizza dei **canali** *(Channel)*, dove il proprietario pubblica messaggi sotto forma di transazioni e gli attori autorizzati che si "iscrivono", possono visualizzarli (in una modalità che può ricordare il funzionamento di YouTube). La proprietà del canale è implementata e protetta in IOTA dal *seed*, che come detto prima non deve essere mai esposto e conservato in modo sicuro**.**

Per pubblicare i messaggi su MAM viene utilizzata la *root* di quest'ultimo. In tutte le modalità infatti, per visualizzare il *payload*[13] della transazione è necessario conoscerne la root, con la quale questo viene criptato e decriptato (sistema di cifratura ad una chiave). Poi, a seconda dell'intenzione, si hanno a disposizione 3 modalità di riservatezza diverse, che variano sulla base di come il messaggio viene criptato e dell'indirizzo al quale la transazione viene pubblicata:

- *Pubblico*: indirizzo = root

I messaggi possono essere visionati da chiunque visualizzi (anche per errore) l'indirizzo.

[13] Chiamato anche "*masked payload*". Consiste nel messaggio effettivo (per esempio dati registrati da un sensore), firma e altre informazioni. Nella modalità *restricted* il messaggio effettivo e le altre informazioni sono criptati con la *side_key*

- *Privato*: indirizzo = hash(root)

I messaggi possono essere visualizzati solo se si conosce la root corretta. Ricordiamo infatti che la *root* non può essere ricavata dall'indirizzo a causa dell'hash. Tale sistema di cifratura infatti rende impossibile il procedimento inverso.

- *Restricted*: indirizzo = hash(root)

I messaggi possono essere letti solo se si dispone della *root* e della SideKey corrette. Infatti, in questo caso il messaggio è sempre pubblicato all'indirizzo che utilizza l'hash della *root*, e inoltre subisce due cicli di cifratura:

3) Il payload viene criptato con la SideKey
4) Il payload subisce il secondo ciclo di cifratura standard con la *root*

4.3.2. Merkle Tree

Ma cos'è la *root* utilizzata per creare l'indirizzo MAM? Per capirlo bisogna prima introdurre il concetto del Merkle Tree:

Un Merkle Tree (albero di Merkle) o Hash Tree (albero degli hash) è una struttura appunto "ad albero". È composto da più nodi:

- degli hash di un blocco di dati, rappresentati dai nodi foglia della struttura, ovvero quelli da cui l'albero è generato e che rappresentano la ramificazione estremo di esso;
- ogni nodo non foglia è un hash creato a partire dalla combinazione dei suoi nodi figli.

Il risultato finale è un singolo hash chiamato *root* (radice) di Merkle. Se ogni nodo ha due figli, l'albero è chiamato Binary Hash Tree.

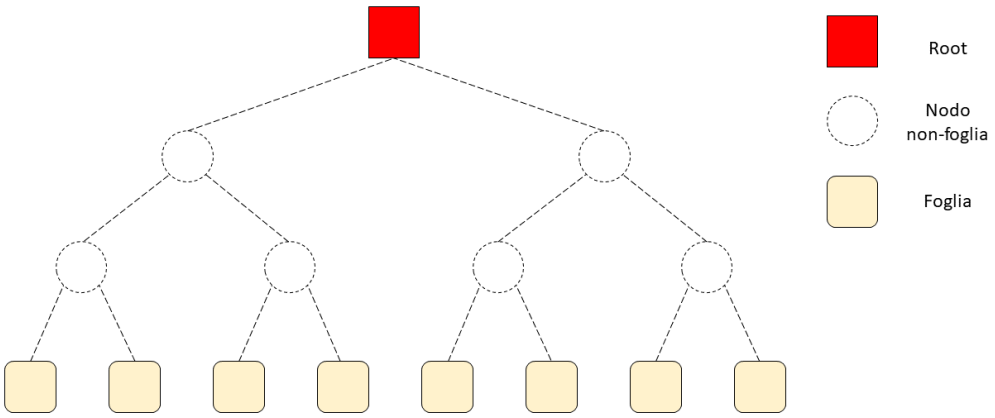


Figura 22-rappresentazione grafica del Merkle Tree

Questa struttura è efficacie per ottenere integrità e validità di un messaggio utilizzando una quantità ridotta di dati, permettendo di risparmiare in termini di memoria.

Innanzi tutto, è importante dire che da uno stesso seed posso generare diversi Merkle Tree e che IOTA utilizza per costruire un canale MAM solamente alberi binari.

Per generare il valore di una root è necessario procedere per step, a partire dai messaggi alla base di uno stesso albero e ne fanno parte. La dimensione dell'albero è definita a priori ed è una potenza di 2. Alla base di un *tree*, le foglie sono numerate con un indice, chiamato *Branch Index,* che ne indica la posizione e che viene utilizzato nel processo di cifratura. Per generare la chiave privata con la quale si esegue l'hashing dei messaggi servono quindi:

- *seed*
- *index*

- *security level* (espresso con un numero da 1 a 3, che regola la lunghezza in trytes della chiave).

In questo modo ottengo un valore, dall'hash del quale ricavo l'indirizzo al quale sarà inviato il messaggio in questione. Eseguendo un'ulteriore operazione di hashing, ottengo finalmente il valore di una delle *leafs* che comporrà il *tree* e che mi porterà ad ottenere il valore che cerco.

Il procedimento consiste nell' eseguire l'hash delle foglie a due a due, fino ad arrivare ad un valore finale, ovvero quello della *root*.

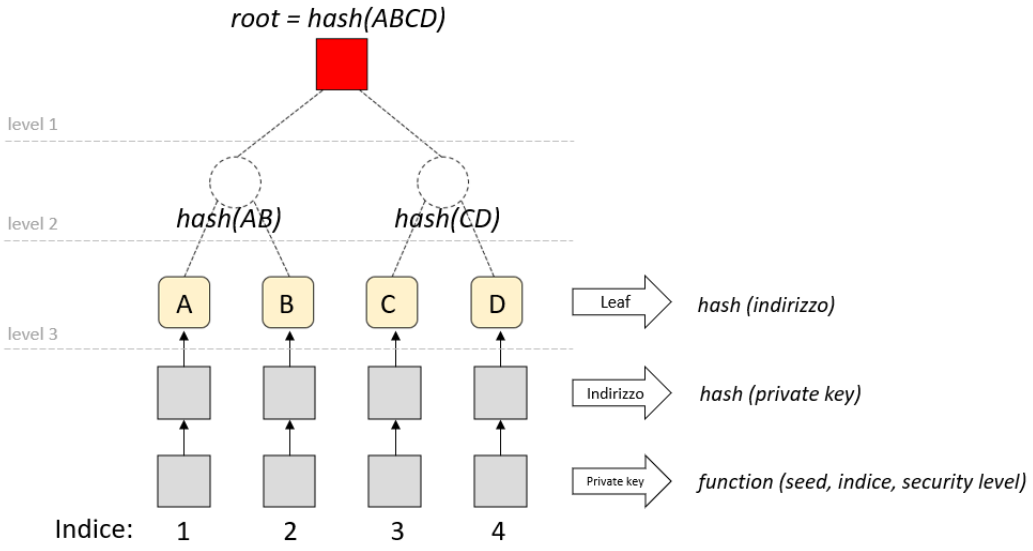


Figura 23-hashing delle leafs e creazione della root

A definire l'inizio e la fine del Merkle Tree ci sono i parametri *start* e *count:*

- *start:* indica l'indice relativo ad un determinato seed al quale si trova la prima *leaf* dell'albero.
- *count:* è il numero di leaf dal quale è composto l'albero (nel caso presentato sotto count = 4, si nota infatti che ogni quattro leaf si genere un nuovo Merkle Tree con una nuova root).

All' aumentare del valore di *count* aumenterà proporzionalmente il numero di livelli dell'albero.

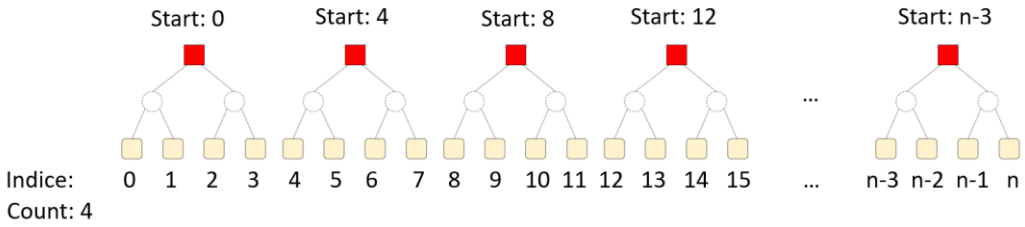


Figura 24-variazione di indice e count

4.3.3. Catena di messaggi

Nel protocollo IOTA, come molti altre DLT *(Distributed Ledger Technology)*, è possibile allegare messaggi di qualsiasi tipo alle transazioni. In questo caso però l'operazione può avvenire senza nessuna transazione.

Senza l'utilizzo di MAM, se si desidera pubblicare dei dati relativi ad esempio ad una misurazione di temperatura attuale con un intervallo di 15 minuti, è necessario trasmettere tutti i messaggi allo stesso indirizzo. Considerando però che il Tangle di IOTA è accessibile pubblicamente, sarebbe facile per gli utenti malintenzionati identificare l'indirizzo e tentare di modificare l'esito delle misure inviandone di fasulle, oppure sovraccaricando il sistema con attacchi spam.

Decidendo dunque di cambiare indirizzo ad ogni transazione, diventa importante tenerne traccia. Questo perché i messaggi si disperderebbero all'interno della rete del Tangle, rendendo molto complicato creare un filo conduttore che colleghi tra loro le transazioni senza conoscerne i singoli indirizzi. Specialmente nel caso di misurazioni inviate con un'alta frequenza, tale operazione diventa però onerosa in termini di archiviazione delle informazioni su cloud.

Grazie alle catene di messaggi che si possono creare con MAM, è possibile proteggere il nostro canale da transazioni spam e liberarsi dalla gestione di indirizzi cumulativi.

MAM pubblica infatti ogni messaggio in un indirizzo diverso, ma con informazioni dettagliate che li connettono direttamente, strutturando la transazione in maniera particolare.

4.3.4. Struttura di un pacchetto MAM

Un pacchetto MAM può essere diviso in due sezioni:

- **sezione della firma** *(Signature Fragment)*
- **sezione MAM.**

L'identità di chi pubblica la transazione è memorizzata nella sezione (citata nel capitolo precedente) del *bundle* della transazione, chiamata *Signature Section*. In questo caso il valore della sezione è creato con la firma digitale anche nel caso in cui la transazione non includa un pagamento, per garantire la proprietà del canale MAM, e quindi anche la sua validità. La *sezione MAM* è dedicata a immagazzinare il vero e proprio messaggio.

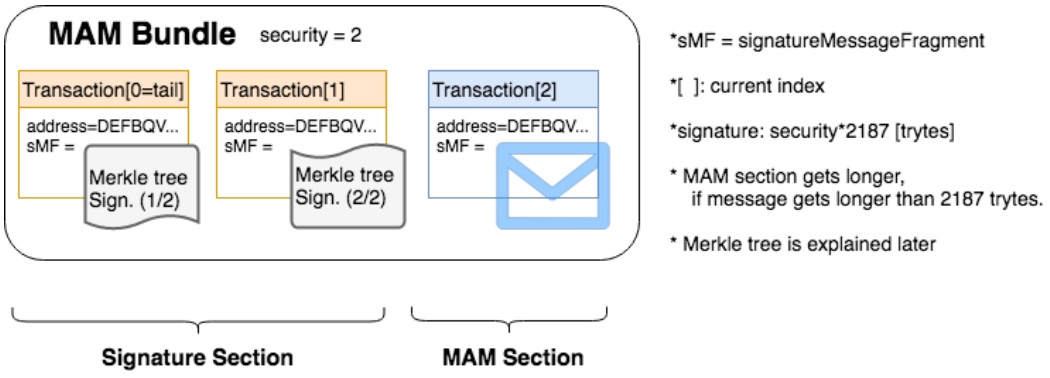


Figura 25-sezioni di un pacchetto MAM

4.3.5. Indirizzi

L'indirizzo dove MAM salva i messaggi su IOTA, è creato a partire dalla *root* della transazione che contiene il messaggio stesso. Nel caso questa venga pubblicata in modalità privata o *restricted*, la root viene sottoposta alla procedura di hashing, rendendo impossibile risalire all'input di origine.

4.3.6. Sequenza di pubblicazione

La nextRoot è un puntatore di connessione al prossimo messaggio. Infatti, andando ad esaminare il contenuto di una transazione inserita all'interno di un canale MAM, notiamo che all'interno sono contenute delle informazioni sul canale. Se prendiamo l'ultima delle transazioni eseguite, noteremo che il campo nextRoot contiene un indirizzo vuoto, che è stato creato al momento della pubblicazione di questa transazione e che rappresenta l'indirizzo al quale verrà inviata la prossima transazione. In questa maniera, è possibile rintracciare tutti i messaggi sulla catena, a partire dalla genesi del canale stesso. Ogni utente sarà in grado di visionare tutte le transazioni a partire dal primo messaggio del canale di cui conosce l'indirizzo. Se si riceve l'accesso a canale avviato, è possibile iniziare a leggere la catena da quel punto in poi e non è possibile risalire e visualizzare i messaggi precedenti.

4.3.7. Proprietà del canale

Consideriamo il seguente caso:

Alice ha appena pubblicato il primo messaggio sul suo canale *restricted* e vuole che Bob ne esamini il contenuto. Quindi, Alice gli passa i valori di root e sideKey. Bob ha rintracciato messaggio generando *l'address* (= hash(root)) di quest'ultimo e decodificandolo con la sideKey.

A questo punto Bob, essendo in possesso della nextRoot, può risalire all' indirizzo seguente. Una volta trovato, realizza che Alice non ha ancora pubblicato il prossimo messaggio, e decide di postare il suo messaggio cifrato con sideKey all'indirizzo hash(nextRoot) (di cui è in possesso), potendo così rubare l'identità di Alice.

Ma grazie al ruolo della firma in MAM, ciò non può succedere.

4.3.8. Sezione MAM

Questa sezione del bundle della transazione contiene il messaggio convertito in tryte e immagazzinato nel campo *message.*

Per eseguire la pubblicazione di un messaggio, l'algoritmo della libreria MAM genera due Merkle Tree, corrispondenti a due indirizzi per la transazione corrente e per quella successiva (nextRoot). Il primo merkle tree (*tree0*) ha due parametri: *start0* (indice da cui parte) e *count* (numero di foglie che comprende). Il secondo (*tree1*) avra dunque *start1* (*start0* + *count*) e *count* (Ogni merkle tree può comunque avere dimensioni differenti.)

All'interno di un singolo Merkel Tree, il campo *branch_index* indica l'indice di ogni singola foglia. Nell'esempio seguente, un albero con *start = 0* e *count = 4* avrà *branch_index = 0, 1, 2, 3.*

Considerando come è strutturato un albero, sappiamo che conoscendo ad esempio il valore della foglia A', per ottenere il valore della root devo conoscere anche B', C', D'. I *siblings* sono i valori degli hashes complementari a quello dato che combinati possono generare il valore della root.

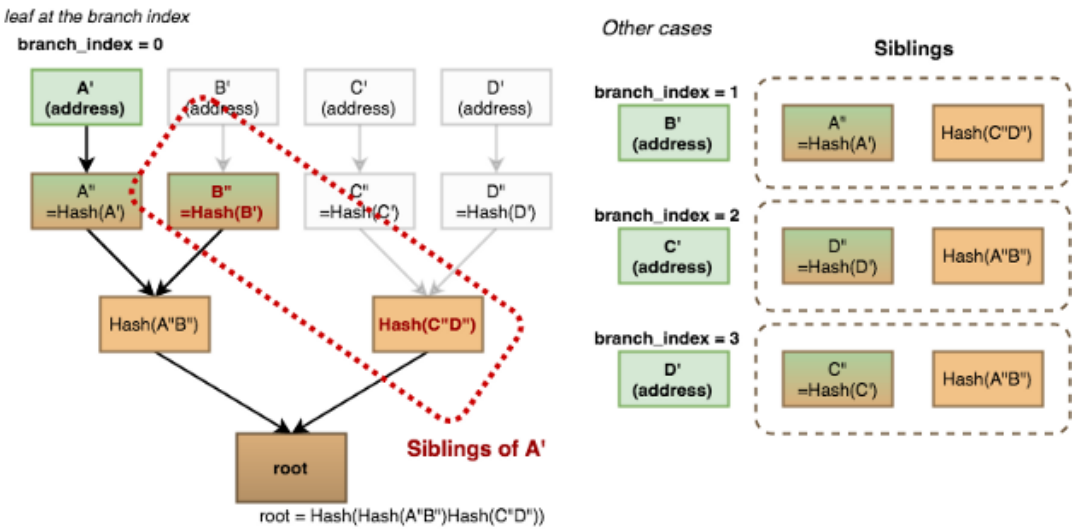


Figura 26-siblings del Merkle Tree

Dall'esempio sopra riportato, si nota che in realtà, nel caso in cui si conosca A', basta sapere il valore di B" e Hash(C"D") per ottenere il valore della root. Questi sono infatti considerati i *siblings* di A' (ai quali in MAM ci si riferisce come *siblings* del *branch_index=0*).

Nella sezione MAM sono dunque presenti i valori di *message, nextRoot, branch_index e Siblings*, criptati con la root (nella modalità pubblica) o con la sideKey (nella modalità Restricted) e immagazzinati all'interno del campo *messageTrytes*.

4.3.9. Sezione della firma (Signature Section)

Come menzionato all'inizio del capitolo, per verificare la validità della sezione MAM l'editore aggiunge la firma *(Signature)* al bundle. La firma è salvata nel signatureFragment e le transazioni che compongono questo "frammento" sono chiamate sezioni della firma *(Signature section)*.

Le funzioni della libreria MAM pongono la firma su un messaggio a partire da una chiave privata creata con seed, branch_index (ovvero l'indice della foglia del merkle tree) e security level scelto. I dati firmati sono rappresentati dal campo messageTrytes del *bundle* MAM.

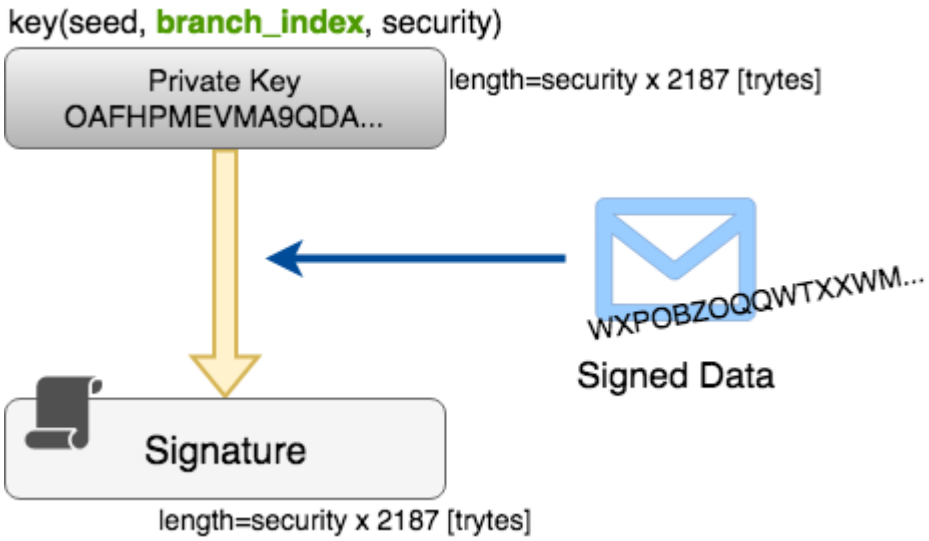


Figura 27-creazione della firma del messaggio

4.3.10. Recuperare i dati (MAM Fetch)

Per recuperare messaggi mascherati servono la *root* e la sideKey nel caso della modalità *restricted*.

Quindi, calcolo l'indirizzo, cerco il bundle contenuto nell'indirizzo e poi decifro il messageTryte trovato nella sezione MAM del bundle, utilizzando la *root* per la modalità pubblica e la sideKey nel caso della modalità *restricted*. Ora dal messaggio decifrato avremo message, nextRoot, branch_index, Siblings.

Ora che abbiamo il messaggio dobbiamo dunque verificarne la validità, a partire dalla *Signature section.* Questa sezione è infatti utilizzata per validare la sezione messageTryte precedentemente citata e marchiata proprio con la firma.

Da questo processo di validazione, si ricava un indirizzo, che rappresenta la foglia di indice = *branch_index* del Merkle Tree che dovrebbe aver generato l'indirizzo della transazione.

Eseguo dunque un ulteriore processo di verifica dell'identità di chi ha creato la transazione, usando questo indirizzo in combinazione con i Siblings che erano immagazzinati nella transazione per calcolare la root dell'albero, che nomino *temp_root.* Se questo valore equivale a quello in cui la transazione è stata immagazzinata, allora MAM riterrà la transazione valida e l'identità sarà considerata confermata. In caso contrario significa che la transazione non è stata pubblicata dal possessore del canale MAM, e dunque la transazione verrà scartata.

(Tale procedimento, come nel caso della pubblicazione, non dovrà ovviamente essere svolto manualmente, ma viene eseguito in maniera automatica delle funzioni della libreria MAM di Iota).

4.4. Sviluppi Futuri

4.4.1. Coordicide

L'uso di un "dispositivo per la verifica delle transazioni" centralizzato, il Coordinator, è stato necessario per garantire la sicurezza durante l'infanzia della rete, rinunciando a parte della decentalizzazione della rete (almeno a livello teorico). La soluzione del Coordicide garantirà che la rete rimanga feeless, preservando al tempo stesso il decentramento e la sicurezza e promuovendo una scalabilità senza precedenti. L'obiettivo è che la rete raggiunga il consenso senza bisogno del Coordinatore, assicurando al tempo stesso che abbia le seguenti caratteristiche:

- *Scalabilità*: Il tasso di transazioni nella rete non è limitato dal protocollo.
- *Sicurezza*: Un aggressore non deve poter influenzare il consenso.
- *Decentralizzazione*: Tutti i nodi onesti possono far parte del processo di consenso.

Le attuali soluzioni DLT possono garantire al massimo due di queste tre caratteristiche contemporaneamente. Questo problema è noto come "scalability trilemma". Una conferma di questo tipo di problema si può vedere in blockchain come Bitcoin, dove si raggiungono sicurezza e decentralizzazione, ma si hanno fondamentali problemi dal punto di vista della scalabilità; per preservare le altre caratteristiche, la blockchain di Bitcoin deve avere dei limiti in termini di dimensioni del blocco, che porrà un limite al tasso di transazioni approvate nell'unità di tempo. IOTA stessa con l'utilizzo del Coordinatore ha dovuto rinunciare in parte alla sua decentralizzazione.

La sola rimozione del Coordinatore non è però sufficiente per ottenere il decentramento. Infatti, il meccanismo di consenso originariamente proposto nel white paper di IOTA richiedeva che la maggior parte delle transazioni provenisse sempre da client onesti. Tuttavia, senza i miners, in IOTA non sussiste il concetto di una potenza di calcolo onesta e costante. Di conseguenza i nodi onesti dovrebbero

inviare un flusso continuo di transazioni, indipendentemente dal fatto che stiano effettivamente utilizzando la rete.

Al centro della soluzione fornita dal Coordicide c'è un meccanismo di voto attraverso il quale i nodi richiedono il parere di altri nodi per decidere quali transazioni debbano essere incluse nel Tangle e quali debbano invece rimanere orfane.

Per rimuovere il Coordinatore, è necessario risolvere una serie di problemi. A causa della complessità della soluzione, il Coordicide rende il DLT di IOTA un software modulare. In questa maniera, ogni modulo può essere sostituito in maniera indipendente qualora nuove ricerche rivelino ulteriori miglioramenti, limitando la stagnazione degli aggiornamenti ed aumentandone la flessibilità.

Di fatto il Coordicide è un meccanismo di consenso migliorato, dove i tempi di approvazione della transazione sono ridotti (pochi secondi), i nodi posseggono una loro identità e votano per risolvere i conflitti chiedendo agli altri nodi lo stato del loro registro.

4.4.2. Smart Contracts.

Gli smart contracts di IOTA possono essere definiti come "*macchine a stato immutabile*":

Macchine a stato: Ogni contratto intelligente ha uno stato ed è allegato al Tangle. Lo stato contiene dati come i saldi dei conti, le condizioni di input e le conseguenze nel tempo. Ogni aggiornamento di stato rappresenta una transizione sul Tangle.

Immutabile: Lo stato e il codice del programma del contratto smart sono entrambi immutabili perché sono memorizzati sul Tangle. Lo stato può essere aggiornato in modo incrementale allegando nuove transazioni.

Il Tangle fornisce una prova verificabile delle transizioni di stato. Ci permette di confidare che le transizioni di stato siano valide e non possano essere corrotte o erronee.

Per facilitare i casi d'uso eseguibili con IOTA, si sta costruendo contratti off-Tangle. Sebbene i contratti on-Chain di Ethereum siano popolari per le loro proprietà, hanno alcuni svantaggi significativi; man mano che la rete cresce, la quantità di elaborazione necessaria per produrre lo stesso risultato aumenta. Questa è un'enorme barriera in termini di scalabilità. Inoltre, il costo per l'esecuzione di questi contratti intelligenti diventa proibitivo.

Gli smart contracts di IOTA sono invece attivati in un contesto localizzato, senza costringere l'intera rete ad eseguirli, evitando impedimenti alla scalabilità della rete.

Ogni contratto intelligente ha un proprietario, che è responsabile di:

Creare il programma di contratto smart e presentarlo alla rete.

- Decidere quanto grande sarà il comitato (il numero N) e selezionare i nodi della rete che ne faranno parte.
- Decidere quanti nodi del comitato devono raggiungere un consenso sugli aggiornamenti dello stato del contratto smart (numero è chiamato *quorum*).
- Definire altri parametri generali di configurazione del contratto smart.

Un proprietario può essere una singola entità, come un'organizzazione o una persona, oppure può essere un insieme decentralizzato di pari, come un consorzio di organizzazioni. In ogni caso, il proprietario gestisce solo l'impostazione e la configurazione del contratto, a seconda del loro contesto e del loro scopo, e non partecipa alla sua gestione. Ad esempio, un contratto smart che gestisce transazioni di alto valore può richiedere un ampio comitato di nodi. Mentre un contratto smart che gestisce microtransazioni può richiedere solo 20-30 nodi nel comitato.

Ci sono molte possibili ragioni per voler creare o gestire uno smart contract. Uno dei motivi è la ricompensa. Anche se le transazioni IOTA sono feeless, i contratti IOTA smart rappresentano un'opportunità di guadagno per le aziende, ad esempio per la copertura dei costi operativi. Sia i proprietari che i nodi del comitato possono

ricevere premi; spetta al proprietario negoziare con i gestori dei nodi del comitato per deciderne il valore minimo.

Un'altra potenziale motivazione per far parte di un comitato è quella di costruirsi una buona reputazione. I proprietari possono scegliere di formare comitati solo di nodi con una buona reputazione. Questo tipo di sistema di reputazione potrebbe creare un mercato aperto per i nodi dei comitati che incoraggi il buon comportamento all'interno della rete.

4.4.3. Streams

IOTA Streams è una nuova versione di MAM, dal quale differisce per alcune caratteristiche che possono essere così riassunte:

- *Framework crittografico*: IOTA Streams non riguarda solo i canali, essendo invece un vero proprio framework per applicazioni crittografiche. La funzionalità dei canali, significativamente estesa rispetto a MAM, è ora solo una delle applicazioni che può essere implementata. Per gli scopi in cui i canali non sono la soluzione ideale, è possibile costruire un'applicazione diversa utilizzando il framework IOTA Streams.
- *Canali*: hanno alcune nuove caratteristiche. Come avveniva anche con MAM, un autore può firmare messaggi e più "abbonati" possono ricevere messaggi. Con Streams però gli abbonati possono ora anche pubblicare anche messaggi non firmati all'interno del canale.
- *Tipi di messaggi*: nei flussi standard di IOTA, i messaggi hanno una struttura basata sull'intestazione. In MAM invece il formato del messaggio era fisso (*public, private, restricted*), e se il vostro caso d'uso avesse richiesto strutture di messaggio diverse, sarebbe stato necessario utilizzare canali diversi per ogni struttura. Ad esempio, si potrebbero desiderare diversi tipi di strutture per il monitoraggio dei messaggi standard e dei messaggi di avviso. In IOTA Streams, è invece possibile pubblicare

entrambi nello stesso canale, e le applicazioni di lettura li distingueranno in base alle loro intestazioni specifiche.

- *Miglioramento nel controllo degli accessi*: in MAM è impossibile fare dei fork del canale per condividere varie informazioni con diverse parti. In IOTA Streams, è possibile applicare un diverso meccanismo di crittografia ad ogni messaggio in base al suo tipo, tutto in un unico canale. Ciò significa che messaggi diversi hanno regole di controllo degli accessi diverse in base al loro tipo, e le diverse parti possono visualizzare solo alle informazioni a cui hanno accesso.
- *Messaggi di collegamento*: in MAM, tutti i messaggi erano indipendenti. Con Streams, i messaggi possono essere collegati l'un l'altro, permettendo l'accesso ad informazioni supplementari sul messaggio corrente.

Inoltre, in IOTA Streams, è possibile modificare i messaggi precedenti in un canale esistente, e mentre il messaggio più vecchio rimane nel Tangle per garantire l'integrità, le applicazioni possono recuperare direttamente solo il messaggio più recente e valido. Questo rende la sostituzione e la modifica di informazioni come le credenziali molto più semplice di prima. È possibile aggiornare le credenziali legate a un'identità digitale, ad esempio, senza dover creare un nuovo canale.

5. PoC: “track and trace” in una Supply Chain con IOTA

5.1. Caso d’Uso

Con questa PoC, si vogliono testare funzionamento e prestazioni di IOTA all’interno di una Supply Chain, per il track and trace dei beni in essa trattati e del loro stato.

Perciò, oltre alla posizione l’obiettivo della demo è quello di misurare alcune caratteristiche delle condizioni in cui il prodotto viene trasportato o conservato, come umidità e temperatura.

Infatti, specialmente nel caso di una supply chain di prodotti freschi come gli alimenti, tali informazioni sono fondamentali per capire in quale punto della catena siano avvenuti eventuali guasti o disattenzioni che hanno portato al deterioramento delle caratteristiche del prodotto o del lotto. In questo modo è possibile risalire facilmente ad un responsabile, evitando dispute a riguardo e avendo la certezza che l’informazione ottenuta sia veritiera.

La soluzione sviluppata ha esaminato un caso semplificato, in cui sono presenti solo due attori fissi all’interno della supply chain. I dati misurati dai sensori sono dunque relativi solo alla permanenza in magazzino del prodotto, e le sue condizioni durante il tragitto sono state appositamente trascurate.

Se una rete così contenuta fosse presente in un caso reale, probabilmente sviluppare un sistema di tracking dei prodotti non sarebbe necessario (a meno che non si tratti di beni estremamente preziosi, delicati o che il rapporto tra le parti sia particolarmente critico).

Considerando però un caso reale odierno, quindi dove la struttura della supply chain è probabilmente molto più intricata e le relazioni molto più complesse dello

scenario ipotetico, è dunque necessario un sistema di tracciamento per consentire la raccolta di informazioni evitando dispute.

Tuttavia, non è ottimale segnalare la custodia dei beni utilizzando un database centralizzato e comune per l'intero ecosistema, in quanto sarebbero rivelate anche a terzi, informazioni di proprietà e riservate su diversi stakeholder, clienti, e catene di distribuzione. Inoltre, sarebbe difficile creare un sistema in grado di tracciare centralmente tutte le possibili interazioni previste per una serie di stakeholder non conosciuti a priori.

L'utilizzo di DLT fornisce in questo caso una soluzione a tale tipo problemi. In particolare, l'utilizzo IOTA per il tracciamento assicura che solo i legittimi proprietari ricevano informazioni sui beni di proprietà, senza che nessuna parte non autorizzata possa raccogliere o accedere a qualsiasi informazione riservata.

Di conseguenza, sono previsti i seguenti benefici:

- Gli attori della supply chain possono rintracciare i beni e verificarne lo stato in qualsiasi momento, prevedendo un eventuale deterioramento di essi;

- Possono provvedere per tempo ad acquistare nuovi beni necessari a soddisfare le loro esigenze di consegna;

- Possono risparmiare tempo e costi associati alla ricerca di quelli mancanti o rovinati;

- Possono risparmiare tempo e costi dovuti ad eventuali dispute;

- I fornitori dei beni possono pianificarne meglio le spedizioni conoscendo esattamente il numero e l'ubicazione dei prodotti “guasti”;

Se uno scenario simile si amplia ulteriormente, includendo nel tracking anche la spedizione internazionale che avviene all’interno dei container, i benefici aumentano ancor di più e si estendono ad eventuali partner logistici che si occupano dello spostamento dei prodotti.

5.2. Struttura della Rete

Con le premesse fatte nel paragrafo precedente, di seguito illustrerò la struttura della PoC sviluppata durante il tirocinio che ho svolto in Blue Reply s.r.l., nella quale è stato testato il funzionamento di IOTA nel contesto di una supply chain. L'architettura sotto illustrata presenta una parte hardware e una software per la gestione dei dati in entrata e l'esecuzione delle transazioni su IOTA tramite canale MAM (per la comunicazione con IOTA sono state utilizzate le API della libreria Javascript di MAM). Queste due parti verranno illustrate separatamente.

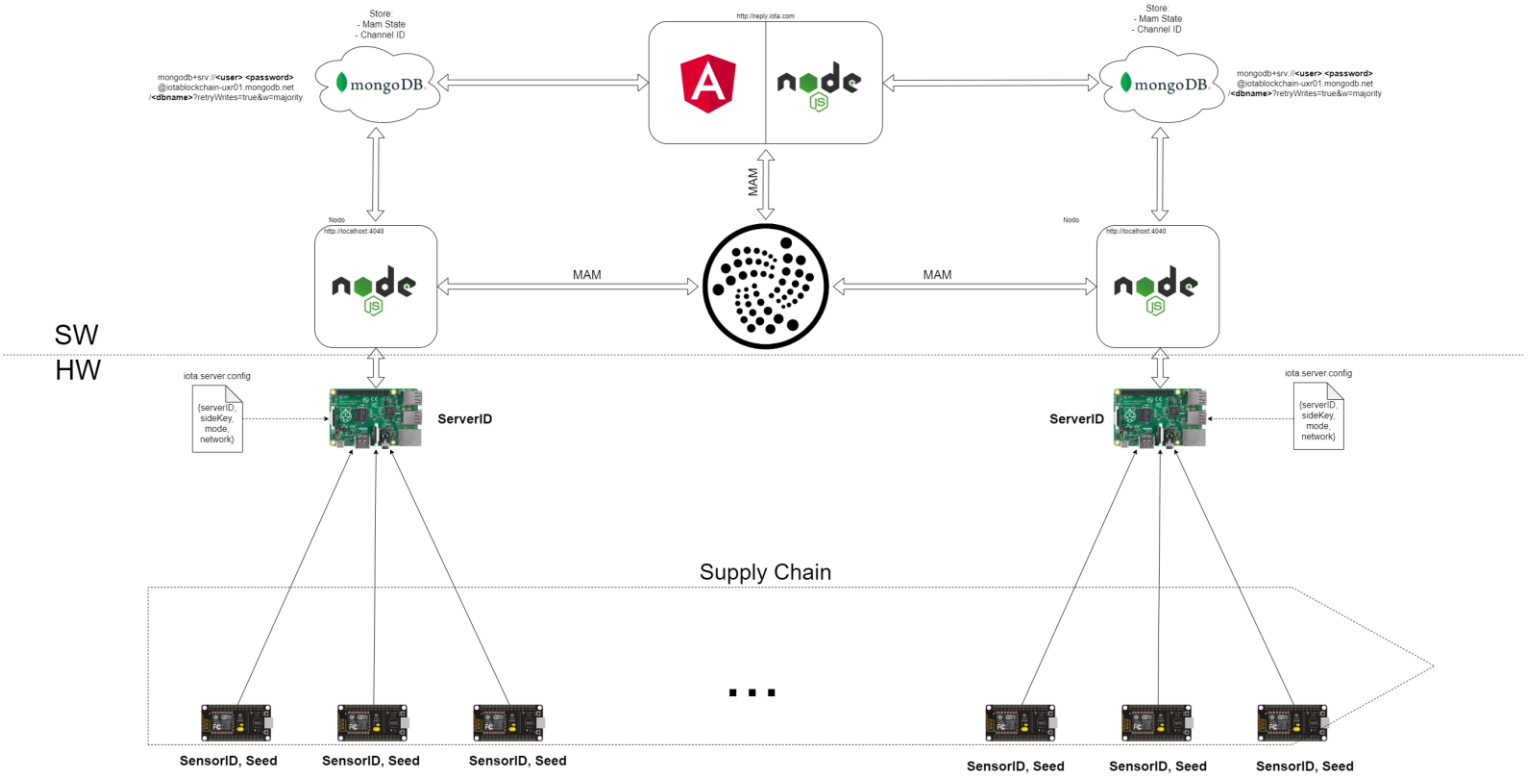


Figura 28-schema dell'architettura utilizzata nella PoC

Innanzitutto, bisogna fare alcune puntualizzazioni sul funzionamento di MAM. Per tenere traccia dello stato del canale, la libreria utilizza un file JSON[14], chiamato Mam State, che ritorna quando la pubblicazione di una transazione va a buon fine.

Di seguito un esempio della struttura del Mam State:

```
{
    "subscribed": [],
    "channel": {
        "side_key": null,
        "mode": "public",
        "next_root": "GNFB...EOAA",
        "security": "2",
        "start": 1,
        "count": 1,
        "next_count": 1,
        "index": 0
    },
    "seed": "OXHU...CMNU"
}
```

Notiamo come queste non siano altro che le informazioni dell'albero relativo al seed espresso nell'ultima proprietà. Sono infatti presenti la sideKey (in questo caso di valore nullo, in quanto il canale è settato nella modalità 'public', come si vede indicato successivamente), la nextRoot del canale, il livello di sicurezza, i parametri del Merkle Tree start e count (qui si vede che count = 1, dunque l'albero è ad una sola foglia e creato a partire dall'indice numero 1), next_count invece indica il count che dovrà essere associato alla prossima transazione, in quanto l'albero per questa deve essere creato in anticipo, e infine index, che è l'indice relativo al singolo albero (in questo caso è per forza zero, visto che l'albero ha una sola foglia).

[14] acronimo di **JavaScript Object Notation**, è un formato adatto all'interscambio di dati fra applicazioni client/server. È basato sul linguaggio JavaScript, riprendendo la notazione letterale degli oggetti, ma non ne dipende.

Tale JSON deve essere dato in input all' API di *publish* per eseguire la transazione seguente, in maniera che ci si riesca a ricongiungere con l'ultima transazione effettuata nel canale. Per fare ciò è dunque necessario salvarne il valore su un file, nel momento in cui lo si riceve. Nella soluzione che andrò a presentare, ho memorizzato questa informazione su un database non relazionale in cloud (MongoDB) invece che su file, per migliorare la funzionalità del codice e ampliare il numero di operazioni eseguibili sul JSON stesso.

Andiamo dunque ad esaminare l'architettura presentata sopra nel dettaglio:

- *Hardware:* in basso si può vedere il flusso della supply chain, rappresentato nel caso della demo dai sensori installati su delle Node ESP8266. Questi in un caso reale potrebbero essere installati sul singolo lotto o in un container, e si muovono da un attore all'altro lungo tutta la supply chain, spostandosi tra i vari magazzini. Questi ultimi sono invece rappresentati da dei RasperryPi (i modelli 3 e 4), sui quali viene eseguito il server dell'applicazione. Questi creano un hotspot nell'area in cui sono situati (SSID: *Iota*), all'interno del quale sono visibili con un indirizzo IP locale standard, che sarà volontariamente settato ugualmente per tutti i Raspberry (in questo caso: *192.168.4.1*). La Node esegue invece uno script che ricerca di default la rete Iota emanata dai Raspberry. In questo modo, nel momento in cui il sensore entra in un magazzino, riconosce l'SSID del Raspberry e invia ad esso le singole misurazioni (*temperatura e umidità*) effettuate da quel momento in poi insieme ad un *seed* univoco, su di esse registrato, tramite delle richieste REST, ad un end-point situato lato server. Qui verrà quindi creato il pacchetto della transazione, da inviare sul canale MAM di proprietà del sensore. Si sarebbe anche potuto immagazzinare il seed direttamente sul server, creando dunque un canale MAM ad esso relativo. Si sarebbe così evitato di esporre il seed ad eventuali intercettazioni nell'atto di invio dalla Node al Raspberry. Tuttavia, così facendo le prestazioni della soluzione sarebbero drasticamente calate, in quanto per visualizzare l'intero percorso svolto da un singolo prodotto nella supply chain si sarebbero dovuti scaricare dal Tangle i dati dei canali relativi a tutti i server in essa

presenti. Bisogna infatti ricordare che con MAM le transazioni vengono recuperate da IOTA in ordine di pubblicazione, a partire dalla prima root di cui si possiede il valore. Detto ciò, nella demo sviluppata, abbiamo deciso di trascurare l'aspetto relativo alla sicurezza del seed per semplicità, sebbene in un caso reale sarebbe da considerarsi un fattore di fondamentale importanza.

- *Software*, la cui struttura è rappresentata nella parte alta dell'immagine ed è costituito da: due componenti server in NodeJS per pubblicare i dati sul Tangle, che vengono eseguiti dai Raspberry, un coppia di componenti Client/Server, che viene invece eseguita su un computer qualsiasi e dai quali sono effettuate principalmente le operazioni di Fetch delle transazioni, un database in cloud su MongoDB e un nodo pubblico, al quale vengono inviate le richieste di publish e fetch, e che comunica con la rete IOTA. Una volta che le misurazioni arrivano sull'endpoint del server, vengono reindirizzate ad una funzione di publish, dove viene creato il pacchetto della transazione utilizzando il seed ricevuto dal sensore. Prima di inviare la transazione, viene recuperato l'ultimo Mam State salvato su MongoDB, che viene utilizzato come riferimento. Quando la publish viene effettuata l'API ritorna il JSON del Mam State. Questo viene dunque salvato su Mongo DB sovrascrivendo quello precedente. Terminato il processo viene inviata una *response* con status 400 al sensore, che viene avvisato della buona riuscita dell'operazione.
  La parte di Client si presenta invece composta da:

Figura 29-mappa dove vengono visualizzati gli spostamenti dei lotti tracciati

- Una mappa che rileva le nuove transazioni e mostra su quale server esse vengono effettuate (in maniera del tutto casuale e immaginaria abbiamo situato i due Raspberry sulla mappa a rappresentare due magazzini in parti diverse del globo). Passando il cursore sulla lista a fianco, vengono visualizzate le ultime posizioni dei sensori.

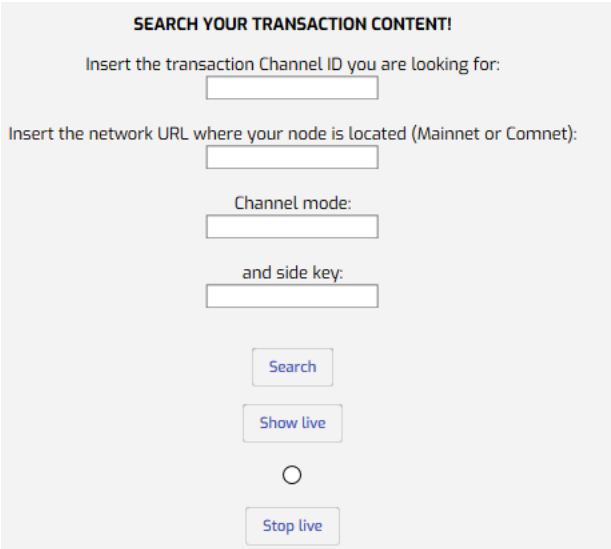


Figura 30-form per esaminare il canale

- Un form da compilare per ricercare una transazione sul Tangle. Per effettuare la ricerca bisogna inserire il Channel ID del canale MAM, l'url della rete IOTA in cui la transazione è stata pubblicata (comnet, devnet, mainnet), la modalità del canale e (nel caso in cui questa sia settata su *restricted*) la side Key, per poter decifrare la transazione. Inoltre, è possibile scegliere se effettuare una ricerca standard, ovvero dove viene eseguita la funzione di fetch una volta sola, oppure la ricerca live, che esegue la stessa funzione in maniera ricorsiva, mostrando eventuali nuove transazioni nella tabella sottostante.

Filter

Message	SensorID	ServerID	Timestamp
{"sensore":"mcu2","messaggio":"MAM Channel update transaction"}	mcu2	node_1	2020-10-14 12:34:07
{"temperature":"28.83","humidity":"39.60"}	mcu2	node_1	2020-10-14 12:51:17
{"temperature":"29.83","humidity":"38.60"}	mcu2	node_1	2020-10-14 12:51:32
{"temperature":"27.83","humidity":"36.60"}	mcu2	node_1	2020-10-14 12:51:43
{"temperature":"28.83","humidity":"35.60"}	mcu2	node_1	2020-10-14 12:52:28

Items per page: 5 1 – 5 of 12

Figura 31-tabella in cui vengono riportate le transazioni presenti sul canale di cui si è eseguita la ricerca

- Qui vengono mostrate le transazioni cercate, filtrabili e ordinabili. Di queste viene riportato il messaggio (un oggetto con i valori di temperatura e umidità), l'ID del sensore che ha eseguito la misura, quello del server che ha creato il pacchetto ed eseguito la transazione e un *timestamp* all'ora europea, che permette di mostrare la sequenzialità delle transazioni. Dal momento che le transazioni sul Comnet di Iota vengono cancellate ogni giorno, prima di iniziare a ricevere transazioni dai sensori è necessario eseguire un'operazione di "*refresh*" del canale. Questa funzionalità è necessaria solo dal momento in cui si usa il network di prova e viene attivata tramite un taso che esegue le transazioni all'interno dei canali vuoti. La prima transazione mostrata in tabella ha infatti contenuto non misurabile e di valore "*MAM channel update transaction*".

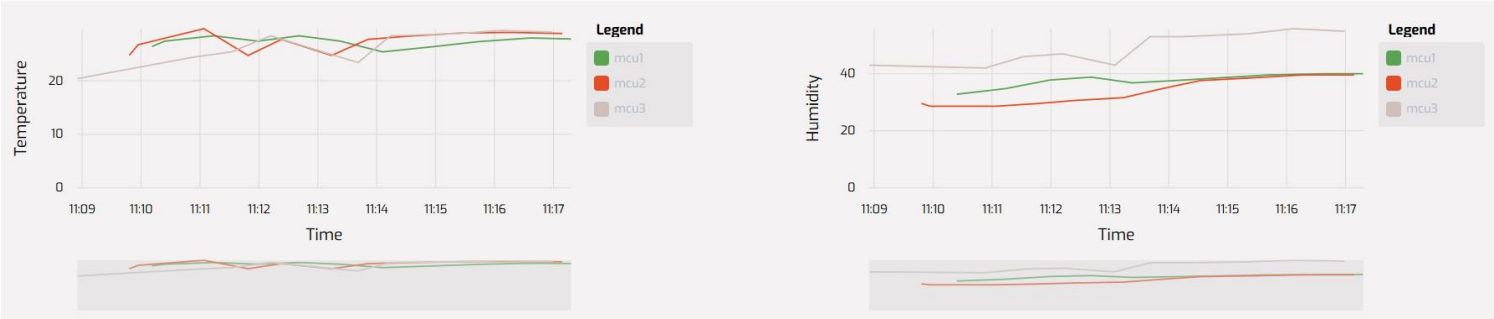


Figura 33-grafici dove sono riportati i valori di temperatura misurati 'live'

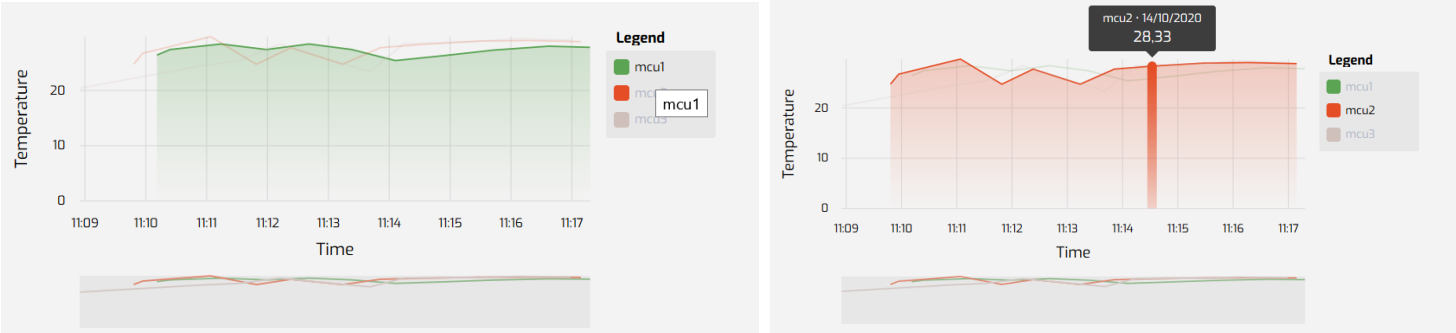


Figura 32-funzionalità dei grafici

- Infine, nei grafici sopra riportati vengono mostrate in live le misurazioni eseguite dai sensori che pubblicano su IOTA, dal momento in cui viene avviato il client della demo in poi, ovvero le misurazioni in tempo reale (le misurazioni riportate in figura sono a fine esemplificativo e non appartengono ad un caso reale). Come mostrato, se si passa il cursore sulla legenda viene evidenziata la curva selezionata; allo stesso modo, se si passa il cursore sopra a curva stessa, si possono visualizzare i valori delle singole misurazioni.

5.3. Sviluppo della Demo

Prima di entrare nel merito di come è stata sviluppata la demo e di quali siano le logiche che guidano le sue funzioni base, di seguito verranno introdotte le principali API utilizzate, per poter comprendere la terminologia che verrà utilizzata in seguito.

5.3.1. MAM API

- ***Init***

Questa funzione inizializza un Mam State nuovo, con campo start = 0, ricevendo in imput tre valori:

```
Mam.init(impostazioni, seed, sicurezza)
```

- impostazioni: oggetto o stringa di configurazione, con l'URL del provider di rete. Serve per indicare a che nodo ci si vuole affidare per fare la PoW.
- seed: stringa del seed che si vuole utilizzare. Se non viene inserito nessun valore, si genera un seed automaticamente.
- sicurezza: livello di sicurezza opzionale che si vuole utilizzare per le chiavi privati. Il valore predefinito è settato a 2.

Il valore ritornato sarà un oggetto con il Mam State inizializzato, da utilizzare in azioni future.

- ***changeMode***

Questa API utiliza l'oggetto di stato creato con *init* e cambia la modalità del canale da quella predefinita (modalità pubblica) alla modalità specificata (pubblica, privata o restricted).

```
Mam.changeMode(state, mode, sidekey)
```

- state: L'oggetto Mam State di cui voglio cambiare la modalità.
- mode: Modalità alla quale voglio passare. Se non inserisco nessun valore, viene utilizzata la modalità pubblica.
- sideKey: la chiave di cifratura richiesta per la modalità restricted.

Viene restituito un nuovo Mam State con la nuova modalità settata.

- ***create***

Crea un payload del messaggio da un Mam State, il messaggio stesso codificato in tryte e la sideKey opzionale.

```
Mam.create(state, message, sideKey)
```

- state: Oggetto del Mam State
- messagge: Stringa contenente il messaggio codificato, che si genera utilizzando la funzione asciiToTrytes dalla libreria @iota/converter di IOTA e passandogli il JSON del Mam State trasformato in stringa (con la funzione di JavaScript *stringify*).
- (opzionale) sideKey: per criptare il messaggio, se la modalità è restricted.

Da qui vengono restituiti:

- stato: Mam State aggiornato
- payload: stringa che rappresenta il payload in trytes
- root: la root relativa al payload, sotto forma di stringa.
- indirizzo: indirizzo in tryte, che verrà utilizzato come destinazione dove pubblicare il payload

- ***attach***

Questa funzione riceve in input il payload precedentemente creato e lo invia al tangle in maniera asincrona.

```
await Mam.attach(payload, address, depth, minWeightMagnitude)
```

- payload in tryte da pubblicare usl tangle
- root: che ho ricevuto da Mam.create ()
- depth: numero opzionale che indica a che altezza inizia la Random Walk per la scelta del tip. Generalmente viene utilizzato un valore di 3, che stabilisce la partenza a 3 milestones nel passato.
- minWeightMagnitude: numero che indica la difficoltà dell'hash. Questo è standard a seconda di quale sia il network sul quale si esegue la pubblicazione. Per il Mainnet è 14 e per devnet e comnet è 9.

Da quest'operazione viene infine restituito un oggetto relativo alla transazione che è stata inserita nel Tangle, all'interno del quale uno die campi è il nuovo Mam State.

- ***fetch***

Questa funzione asincrona esegue una ricerca di tutte le transazioni del canale a partire da una root nota.

Può essere utilizzata in due modi: Senza l'uso di una callback la funzione legge l'intero canale prima di restituire un risultato. Con la verranno restitiuiti i dati attraverso di essa fino al raggiungimento nextroot del Mam State attuale, ovvero quando non sono presento nuove transazioni sul canale.

```
await Mam.fetch(root, mode, sidekey, callback, limit)
```

- root: Stringa della root usata come punto d'ingresso ad un canale (Channel ID).
- mode: modalità del canale (pubblico, privato o restriced)
- sideKey: se il canale è restricted, è indispensabile per decifrare il messaggio.

- callback: opzionale. Se nullo la funzione restituisce le transazioni dell'intero canale
- limit: valore opzionale che indica il numero massimo di transazioni da restituire con la fetch (a partire dal Channel ID)

Restituisce:

- nextRoot: Stringa che corrisponde alla root della prossima transazione; di conseguenza non conterrà ancora nessun valore, ed è il punto in cui la ricerca si interrompe.
- messaggi: lista in tryte di tutti i messaggi contenuti nel canale.

- ***fetchSingle***

```
await Mam.fetchSingle(root, mode, sidekey)
```

Opera nello stesso modo della fetch standard, ma ha un valore di limit settato di default a 1.

5.3.2. Publish e Fetch

Di seguito sono riportati i diagrammi di sequenza delle due operazioni principali eseguite nella demo sviluppata: pubblicazione (Publish) e ricerca (Fetch) delle transazioni presenti all'interno del canale. Questo processo è specifico per come è stata strutturata la PoC in analisi (vedi paragrafo 5.3), quindi non ha valore assoluto.

Le operazioni analizzate coinvolgono 6 "attori" che sono attivi nel processo: uno user che comunica dal Front-End dell'applicazione, i sensori che eseguono le misurazioni, il server con lo script user.route che comunica con il tangle inviando e richiedendo transazioni, le collection sul database realtivo alla demo salvate in cloud con MongoDB, il canale MAM creato con le API della rispettiva libreria

sopra riportate e infine il Tangle di IOTA con il quale si comunica passando attraverso un nodo pubblico. Andiamo dunque a descrivere i due processi in ordine.

- ***Publish***

Il primo processo è volto a realizzare una transazione e parte da una richiesta REST che esegue il sensore, inoltrando la misurazione e il seed al server eseguito sul Rasperry di riferimento. Successivamente con la funzione *init* viene inizializzato un Mam State e una volta creato si esegue la transizione alla modalità restricted (utilizzando *changeMode*), passando il valore della sideKey conservata all'interno del server. Dopodiché si esegue una verifica sul Mam State, confrontandolo con quelli già presenti su MongoDB, per verificare se esiste un canale relativo al seed su di esso indicato; per fare ciò si paragona il seed attuale con tutti quelli presenti su Mongo, e in caso affermativo si verifica che il relativo Mam State presenti un valore al campo nextRoot (per verificare che sul canale in esame sia già stata eseguita almeno una transazione; se il valore è "*null*" vuol dire che ciò non è verificato). Nel caso in cui esista un riscontro positivo relativo al canale corrente, il Mam State inizializzato viene sostituito con quello recuperato dal database, per allacciare la transazione all'ultimo messaggio precedentemente pubblicato.

A questo punto, con la funzione *create* per creare il payload della transazione e successivamente *attach* per pubblicarlo sul Tangle.

Verrà dunque restituito l'oggetto relativo alla transazione, dal quale si estrapola il nuovo Mam State da salvare su MongoDB, sovrascrivendo la versione precedente. In questo modo la transazione successiva partirà dalle nuove informazioni in esso contenute.

Infine, il nuovo Mam State viene restituito al sensore con un codice 400 (operazione andata a buon fine) e la richiesta http viene chiusa.

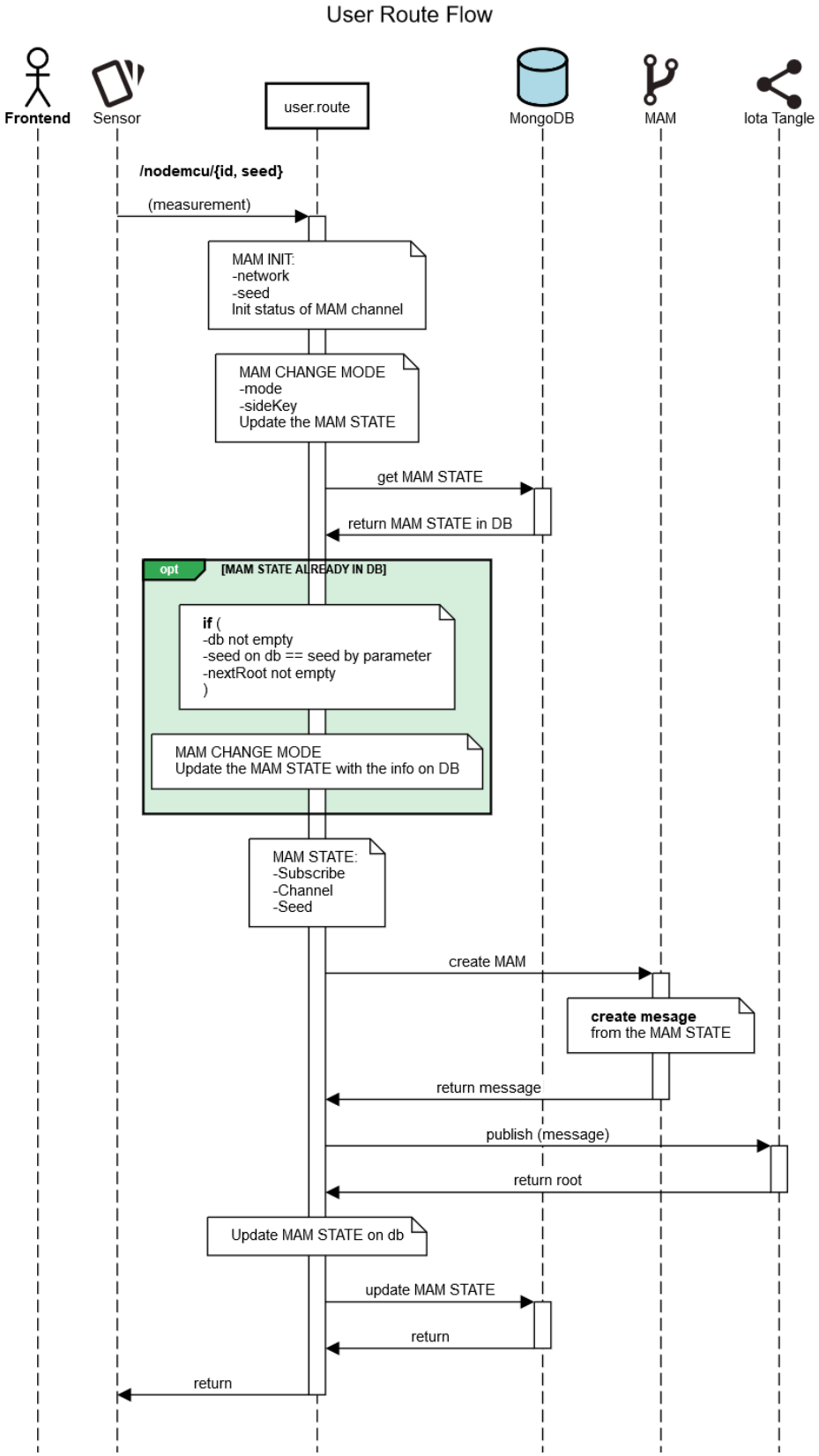


Figura 34 - diagramma di sequenza dell'operazione di Publish.

- ***Fetch***

In questo caso invece il processo ha origine dalla volontà di uno user (es. dipendente di un'azienda che fa parte della supply chain), che da front-end inserisce le informazioni del canale di cui vuole visualizzare lo storico. Per fare ciò deve inserire la root, il network di riferimento, la modalità del canale e la sideKey ad esso relativa (come visto nel paragrafo 5.2). Da qui verrà inviata una richiesta REST al server tramite protocollo http, in questo caso eseguito sullo stesso computer del client. Qui si eseguirà la ricerca tramite MAM, fornendo all'API i valori di Channel ID (root), sideKey e modalità. Il risultato della ricerca verrà salvato su una lista che a sua volta sarà restituita come risposta alla chiamata http. Bisogna però specificare che, nella soluzione proposta, per eseguire la ricerca è stata utilizzata la *fetchSingle* anziché la *fetch* standard, inserendola in un ciclo while. Questo per migliorare le prestazioni della demo, in quanto la versione standard dell'API sovraccarica il server. Perciò, viene eseguita la ricerca dei messaggi uno ad uno recuperando la root dal risultato di quella precedente, fino a quando non viene restituito un valore *null*, che segnala la fine del canale.

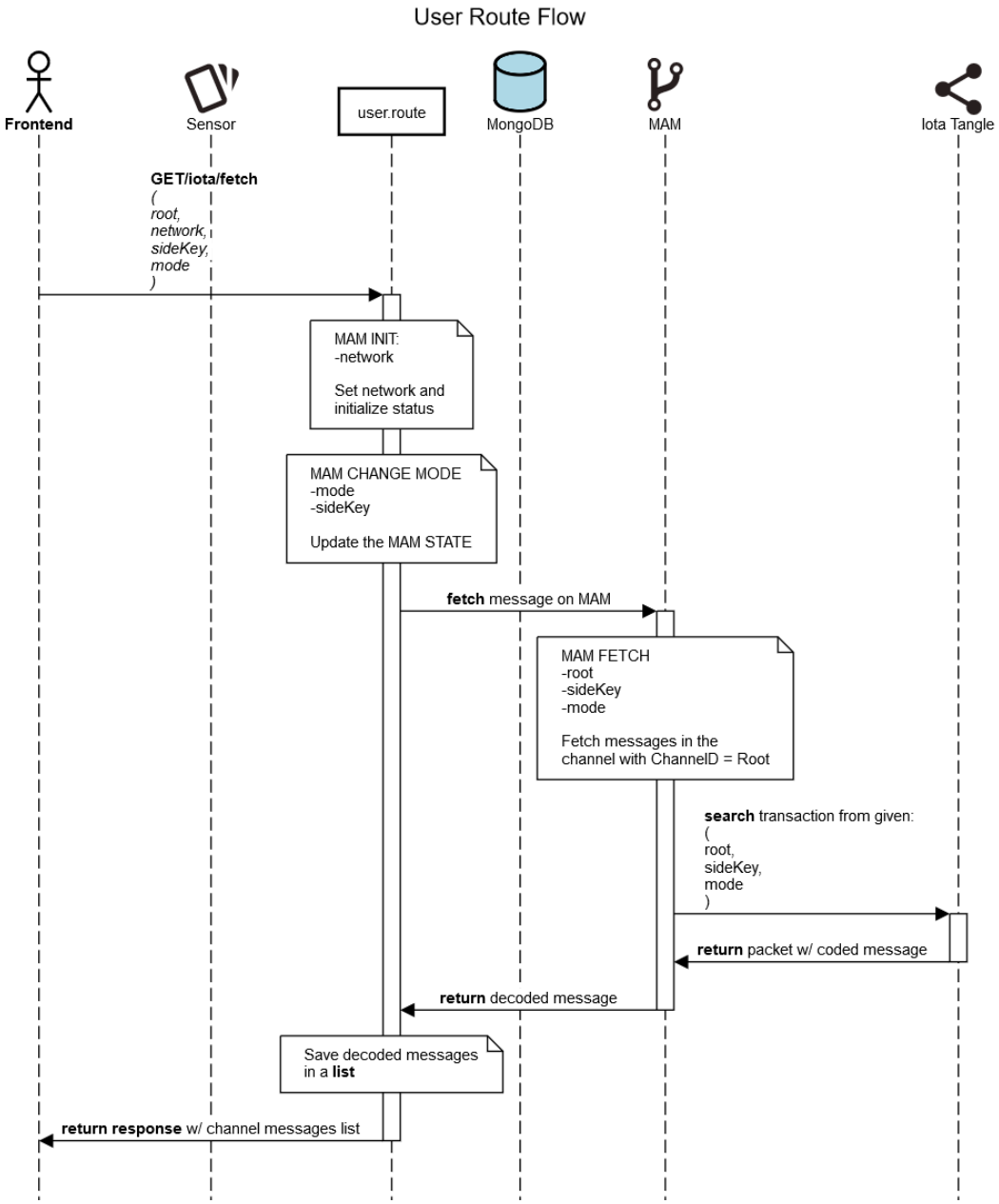


Figura 35 - diagramma di sequenza dell'operazione di Fetch

5.4. Analisi delle Prestazioni

Dopo aver finalizzato lo sviluppo della PoC descritta nei paragrafi precedenti, le ultime due settimane di permanenza in Blue Reply sono state dedicate allo svolgimento di test e all'analisi di quest'ultimo sulla soluzione. Le osservazioni sono state svolte al fine di mettere alla prova le prestazioni delle API del MAM (masked Autenticated Messaging) di IOTA, in particolar modo le funzioni di Publish e Fetch, sul e dal Tangle. Le transazioni sono state eseguite all'interno della rete di prova di IOTA, supportata dai nodi della Community (Comnet).

Con il primo grafico è stato analizzato il caso di studio più semplice, per testare il comportamento della Publish su un periodo di tempo abbastanza lungo (4h c.a), con un intervallo tra richieste di 20sec. Lo scopo è quello anche di verificare se la differenza di prezzo tra il primo ed il secondo è necessaria per avere migliori prestazioni, oppure se è sufficiente un modello meno evoluto per svolgere il ruolo di nodo in una rete di questo tipo. I costi per le componenti hardware della rete (i due RaspberryPi come server e delle Node MCU ESP8266 come sensori) ammontano a:

Componente	***Prezzo***
RaspberryPi 3 Model B V1.2	35,00€
RaspberryPi 4 Model B	60,00€
Node MCU ESP8266	8,00€

Da qui è emerso l'andamento con tendenza lievemente crescente dell' intervallo richiesto. Inoltre si nota anche graficamente come il server 'node_2' (RaspberryPi 3 Model B V1.2) abbia prestazioni della CPU inferiori a node_1 (RaspberryPi 4 Model B), fattore che si rispecchia in un tempo di Publish medio maggiore e in una maggiore variabilità dello stesso.

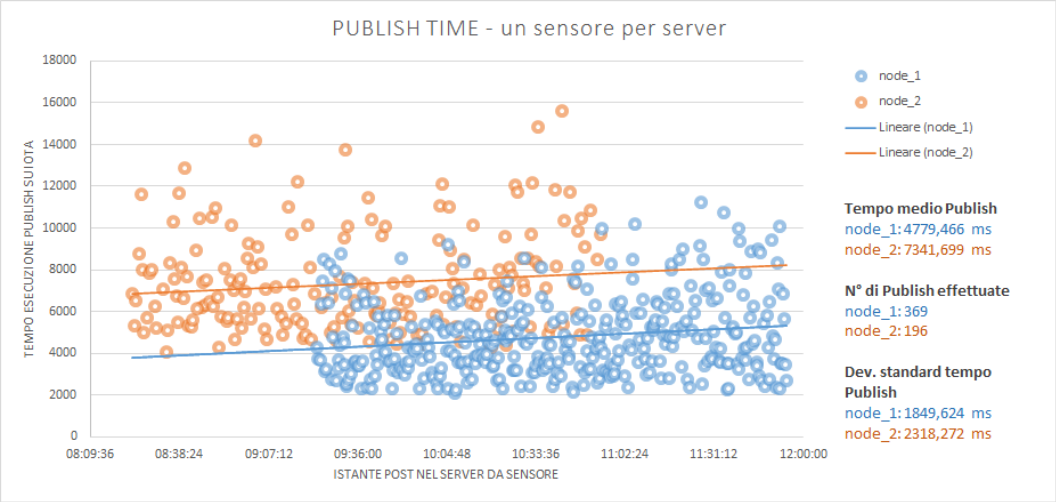


Grafico 2-publish time dei due server Raspberry

In ogni caso entrambe le misurazioni presentano un andamento leggermente crescente, dovuto probabilmente al surriscaldarsi della CPU (questo fattore lo analizzeremo in seguito). Scendendo più nel dettaglio, si nota che la distribuzione dei tempi di Publish, segue l'andamento della curva di Erlang, allo stesso modo dal probability plot. Quest' ultimo è una tecnica grafica per valutare se un set di dati segue o meno una data distribuzione, rispetto a quella standard della distribuzione normale, per la quale i punti dovrebbero formare approssimativamente una linea retta. In questo caso il grafico non segue una retta, ma ha una froma irregolare, che rimanda alla Erlang.

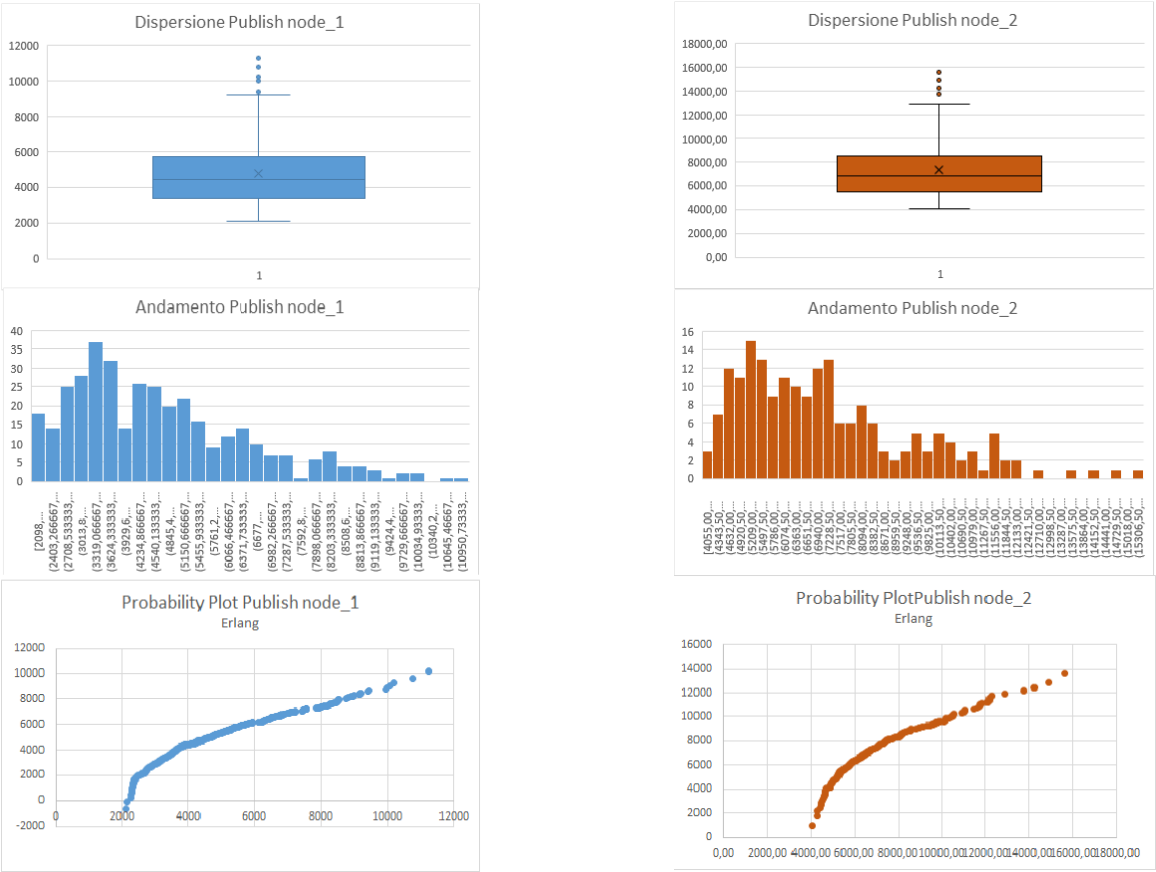
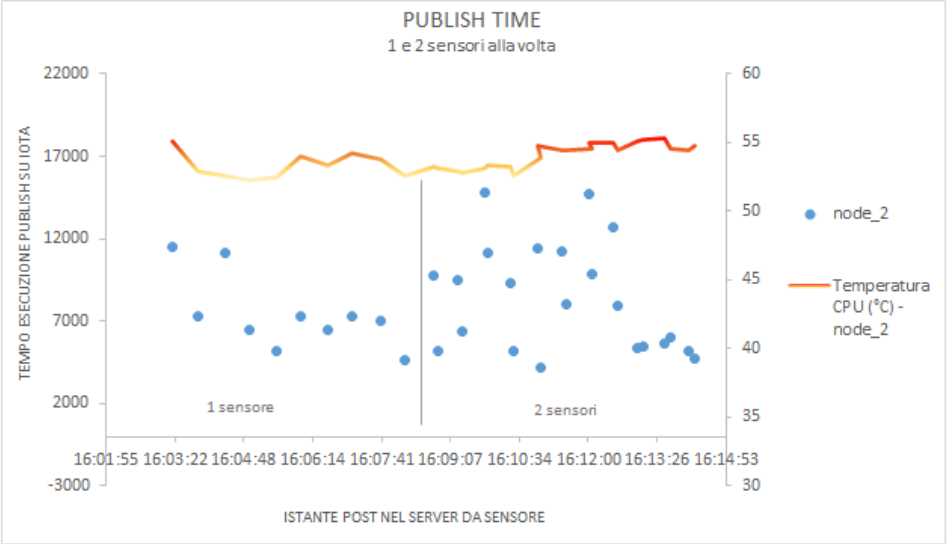


Grafico 3-distribuzione dei Publish Time sui due server

Per mettere i server sotto stress, nelle seguenti analisi abbiamo invece testato la solidità della rete prima aumentando il numero di sensori che inviano richieste REST in contemporanea, e successivamente diminuendo l'intervallo di tempo che intercorre tra una richiesta e l'altra. Detto ciò, bisogna tenere conto che l'esito di una publish è limitato dal fatto che dopo ogni transazione deve essere aggiornato il Mam State del canale su cui viene pubblicata, situato all'interno di una collection di MongoDB, che servirà per la transazione successiva. Se nel momento di una nuova richiesta quella precedente non ha ancora terminato l'esecuzione, quest'ultima rimarra slegata dalla catena del canale MAM, andando così persa.



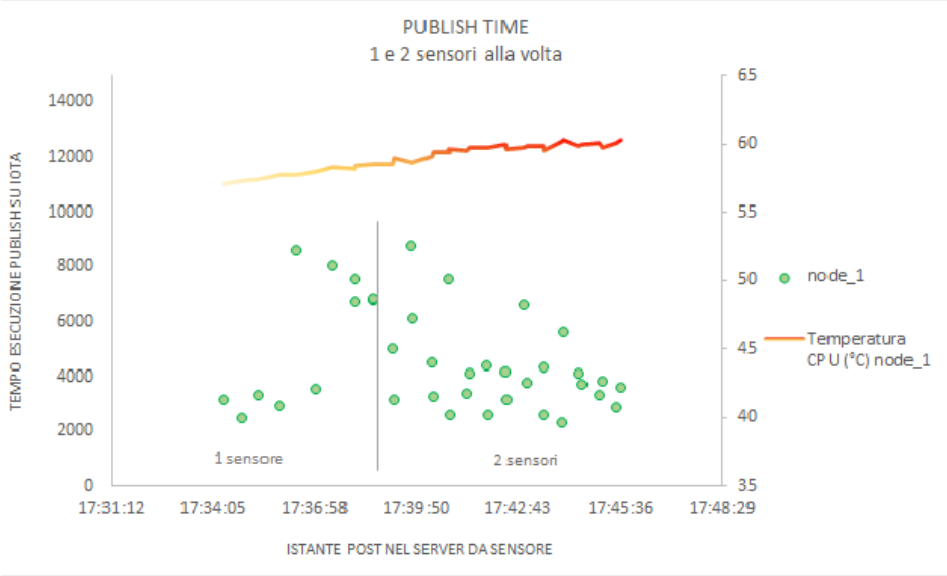
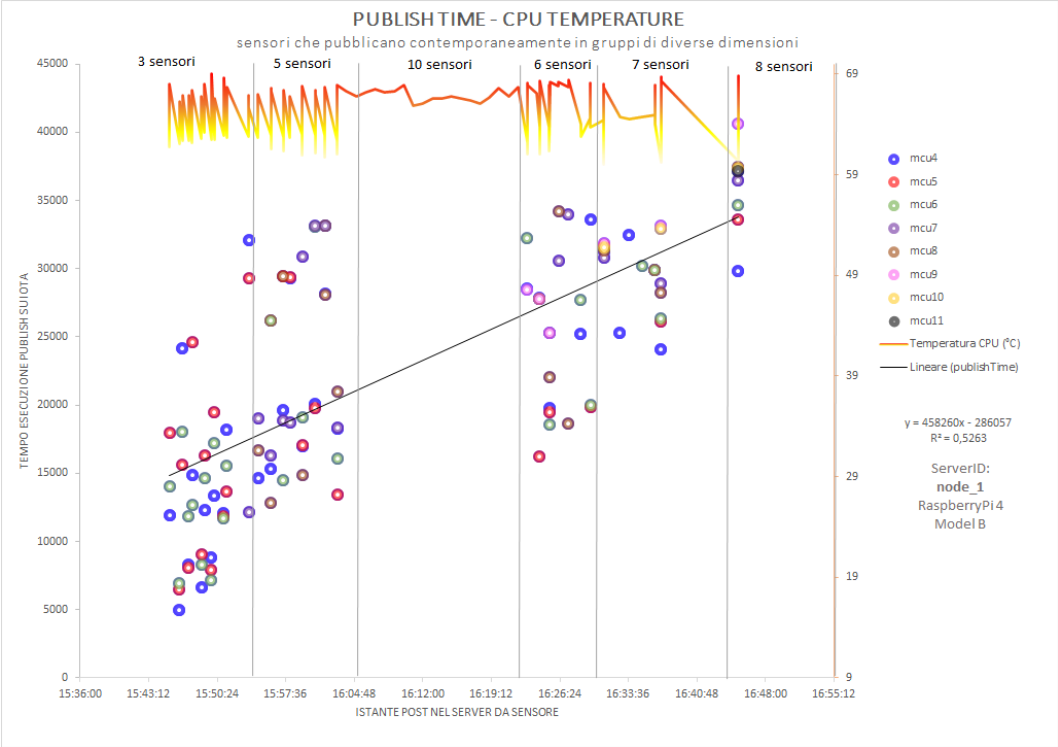


Grafico 4-publish Time in relazione alla temperatura della CPU (1 e 2 sensori attivi)

Il primo passo è stato quello di vedere come varia l'attività aggiungendo un secondo sensore in parallelo al primo, si nota che la temperatura della CPU tende ad essere più alta con un maggior numero di richieste; questo fattore è evidente in particolar modo per il server node_1, dovuto alle caratteristiche del RaspberryPi 4, che richiedono uno sforzo molto maggiore, ma che riesce a soddisfare in maniera più efficiente le richieste.



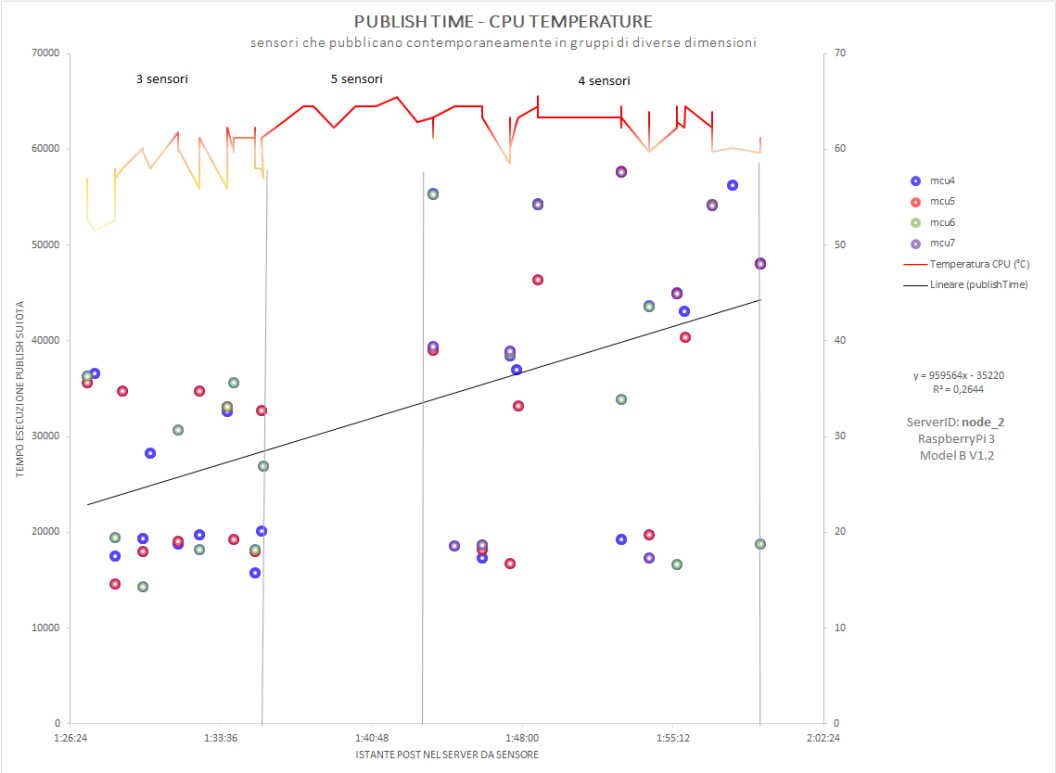


Grafico 5-publish time in relazione alla CPU (molteplici sensori attivi in contemporanea)

Come si può osservare dai precedenti grafici, l'andamento della temperatura della CPU è lo stesso (crescente) al crescere delle richieste. Inoltre risaltano i limiti dei due server; crescendo da 3 a 5 e 10 richieste contemporanee, una volta trovato un muro oltre il quale il server non è più in grado di elaborare transazioni, è stata posta attenzione anche sui casi intermedi. Si è così riscontrato che il limite per node_1 (Raspberry 4) è di 8 sensori e per node_2 (Raspberry 3) è di 4.

N.B: il gap presente in entrambi i grafici è dovuto al fatto che nessuna transazione è stata registrata.

È dunque dimostrata la necessità di un dispositivo più performante, in relazione alla versione attuale della piattaforma. Questo fatto rende dunque necessaria una spesa maggiore per i server, ipoteticamente dislocati lungo la supply chain all'interno dei magazzini.

Sono invece mostrati nella tabella seguente i risultati ottenuti testando il limite della soluzione al diminuire degli intervalli tra una publish e l'altra (un solo sensore). Si conferma il fatto che la risposta della CPU al maggior numero di richieste (siano esse da parte di uno o più sensori) coincide con un aumento della sua temperatura. Inoltre, rispettivamente per i casi con intervalli di 20,15,10 e 5 secondi la percentuale di transazioni riuscite è del 100%,98%,77% e 13%. Questo risultato è in parte in contrasto con l'osservazione iniziale del tempo medio di una publish, che per node_1 è di circa 4,7 sec. Con questo dato alla mano ci aspetteremo un comportamento affidabile della soluzione fino a intervalli di 10sec, dove però si nota già un calo nelle performance. Questo è dovuto al fatoto che il tempo medio è anche correlato all' intensità con la quale vengono sottoposte richieste al server.

PUBLISH PERFORMANCE			
n° sensori fisso (1), intervallo di Publish variabile (20sec, 15sec, 10sec, 5sec)			
Server: node_1 (RaspberryPi 4 Model B)			
Intervallo tra le POST dei sensori sul server	Tot. Tentativi Publish	Publish riuscite sul canale MAM di IOTA	Temperatura media della CPU (°C)
20 sec	100	100	58, 26
15 sec	100	98	59, 90
10 sec	100	77	62, 04
5 sec	100	13	66, 39

Tabella 4-Performance della Publish al variare dell'intervallo tra le richieste

Infine è stata eseguita un'ultima analisi su node_1, con l'obiettivo di indagare l'andamento del tempo di Fetch al crescere delle transazioni da ricercare. In questo caso l'analisi è stata effettuata in maniera indipendente dalla temperatura delle CPU, in quanto la fetch viene eseguita da PC, eseguendo una build completa del server e del front-end (mentre sui Raspberry è eseguito solo il server).

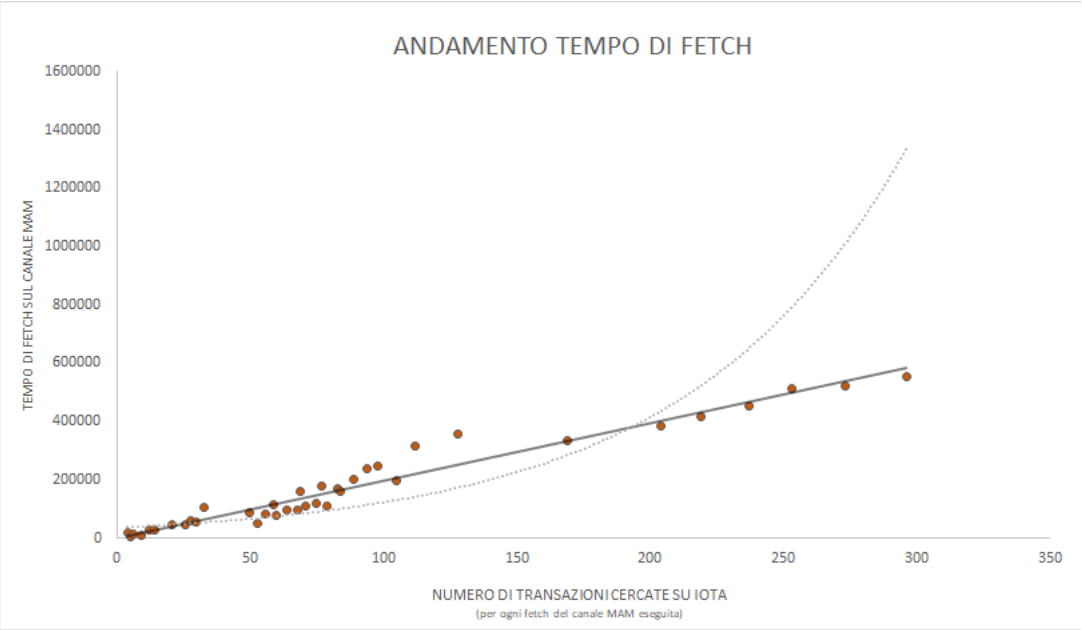


Grafico 6-andamento del tempo di Fetch

Dal grafico si nota che l'andamento è crescente, come ci si aspettava. Questo perché le transazioni sono legate una ad una e per risalire all'ultima, IOTA deve percorrere il 'path' completo del canale. Nonostante ciò, tale rapporto cresce però in maniera lineare e non esponenziale.

6. Conclusioni

Come osservato da Marco Cantamessa e Francesca Montagna all'interno del libro "Management of Innovation and Product Development", è possibile collegare scoperta, invenzione e innovazione / sviluppo si un prodotto come in figura.

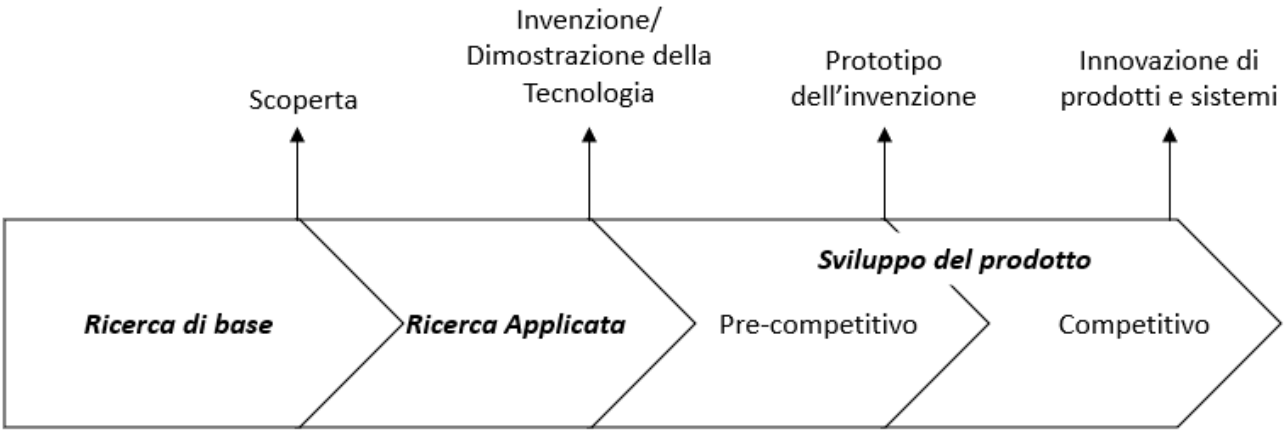


Figura 36-processo lineare di innovazione

Facendo riferimento a questo modello lineare del processo di innovazione (seppur sia semplificato), si possono ripercorrere le fasi che sono state affrontate durante lo sviluppo di questa tesi. L'attività responsabile della scoperta (di IOTA in questo caso) è chiamata ricerca di base, dove per base si intende che non è diretta ad una specifica applicazione. In questa fase mi sono dedicato allo sviluppo della ricerca dello stato dell'arte sul funzionamento e sui vari framework esistenti di DLT. La seconda parte della tesi si è invece dedicata all'attività che porta all'invenzione di una nuova tecnologia, chiamata "ricerca applicata" (seppur in questo caso si trattasse della semplice applicazione di una tecnologia). Il risultato pratico di questa fase è generalmente un "dimostratore", che è un prodotto il cui scopo è semplicemente quello di mostrare che l'invenzione è tecnicamente fattibile. I

dimostratori sono infatti di solito molto lontani dal commerciale prodotti. In questa fase sono stati eseguiti i primi script di prova della tecnologia che, differentemente da ciò che avviene di solito, erano contestuali al campo di applicazione, ovvero il monitoraggio di una Supply Chain. Infine, la ricerca si è conclusa con lo sviluppo di un prototipo dell'"invenzione", senza portarsi alla fase competitiva e di analisi del mercato.

La tecnologia è però soggetta in realtà a fasi distinte e alternate di evoluzione e di progresso. Se si sceglie un'industria e si identifica un indicatore di performance

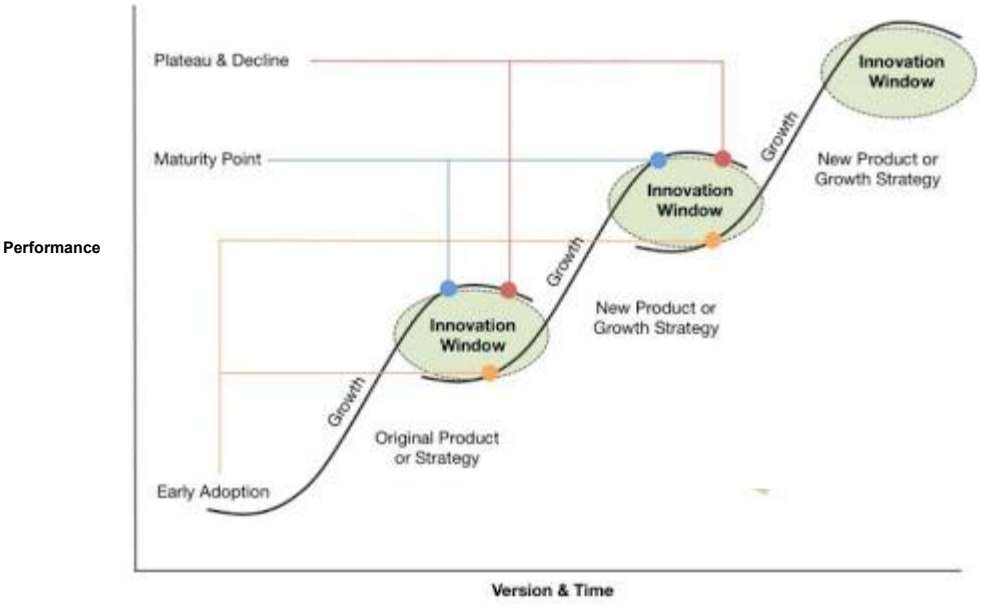


Figura 37-processo ciclico di innovazione di una tecnologia (curve ad S)

rilevante per i suoi, la sua evoluzione non procederà in linea retta, ma seguirà una sequenza di curve a S, come mostrato nel seguente grafico.

Da questo modello emerge che, al nascere di una tecnologia, le prestazioni sono di livello solitamente abbastanza basso. Nelle fasi successive però si presenta un'evoluzione che continua fino al raggiungimento di un certo grado di maturità.

Questo rappresenta il limite tecnologico, un livello di prestazioni che non può essere superato a causa di limitazioni intrinseche. Raggiunto il limite, per migliorare

l'efficienza di alcuni prodotti è necessario abbracciare nuove soluzioni tecniche. Nel fare ciò, avverrà una selezione tra un certo numero di nuove tecnologie candidate. Generalmente, solo una tecnologia si rivelerà adatta ad iniziare una nuova fase del ciclo, sostituendo quella precedente.

In questo processo, il livello di attrazione della domanda sarà uno dei fattori determinanti, mettendo a punto e migliorando la tecnologia in questione in base alle esigenze del cliente. In alcuni casi, quando le aziende sono costrette a "guardarsi intorno" per trovare nuove soluzioni che possano soddisfare i loro bisogni, vi sarà invece un "approccio push" all'innovazione.

L'IBM Institute for Business Value ha intervistato circa 3000 dirigenti d'azienda, classificandoli in tre gruppi differenti sulla base di quanto siano coinvolti nello sviluppo di soluzioni con l'utilizzo di blockchain o DLT. Tali categorie sono quelle di *esploratori*, *investigatori* e *passivi*. Gli esploratori (*early adopters*), attivamente coinvolti nello sviluppo di blockchain, rappresentano l'8% degli intervistati nei vari settori industriali. Gli investigatori (*early majority*), ne stanno invece valutando l'utilizzo, ma non sono pronti a implementare soluzioni reali, e costituiscono il 25% degli intervistati. I restanti (67%) adottano invece un comportamento passivo (*late majority*), non prendendo nemmeno in considerazione tali soluzioni per il momento.

Ciò che è emerso nella precedente analisi delle prestazioni del prototipo, delinea chiaramente alcune caratteristiche che indirizzano verso il tipo di utilizzo che deve essere fatto della PoC sviluppata. Il valore alto dei tempi di Publish e Fetch sottolineano i limiti della soluzione allo stato attuale e mostrano come questa non possa reggere il confronto con le prestazioni di un database tradizionale. Il vantaggio maggiore fornito dall'uso di questa piattaforma contestualmente al tracking di carichi o lotti, è l'informazione sicura, immutabile e in tempo reale relativa al valore misurato di temperatura e umidità. Questo rispecchia alla perfezione le caratteristiche intrinseche di molte piattaforme DLT, che come abbiamo detto all'inizio della tesi devono essere in grado di assicurare:

- Decentralizzazione
- Trasparenza
- Sicurezza (basata su tecniche crittografiche)
- Immutabilità (del registro)
- Consenso
- Responsabilità (tracciabilità delle transazioni)
- Programmabilità

Nonostante ciò, nell'utilizzo che ho fatto della tecnologia e per quanto ho potuto riscontrare durante lo sviluppo dell'applicazione, IOTA non si dimostra ancora maturo a tal punto da poter essere consigliato e adottato all'interno di un caso reale, a meno che non si abbiano fini sperimentali. In particolare, per quel che riguarda le librerie MAM, si riscontrano delle API ancora acerbe, che non permettono una completa maneggevolezza del canale. Ad esempio, risulta attualmente impossibile eseguire un'operazione di fork del canale, al fine di poter permettere la granularizzazione delle informazioni e di poter regolare l'accesso ad esse (ad esempio a seconda del ruolo che si ricopre all'interno della supply chain). È inoltre possibile operare esclusivamente con Merkle Tree ad una singola foglia; con un albero a più foglie è possibile eseguire la Publish, ma poi la funzione di Fetch non permette di accedere a tutti i messaggi, parte dei quali risultano quindi inutilizzabili. Nonostante ciò, di recente IOTA sta programmando il lancio di Streams, il sostituto di MAM che dovrebbe risolvere gran parte di questi problemi, andando anche oltre (vedi pag.93).

Bisogna però dire che già allo stato attuale si potrebbero apportare dei miglioramenti strutturali alla stessa soluzione che è stata presentata. Dopo aver analizzato i limiti massimi della PoC all'interno dell'analisi delle prestazioni, è stato ipotizzato che eseguendo la soluzione su dei container[15] software (ad esempio

[15] è una forma di server virtualizzato a livello del sistema operativo. Invece di creare una istanza virtuale di tutto un server fisico (processore, storage, connessioni di rete, sistema operativo…) come accade per le macchine virtuali, nell'**approccio IT a container** si attiva una istanza virtuale **solo dello spazio utente**, quindi essenzialmente dell'ambiente di esecuzione delle applicazioni.

tramite Docker) ed eseguendo una clusterizzazione[16], l'applicazione sarebbe stata in grado di gestire in maniera ottimizzata un maggior numero di richieste contemporaneamente. Inoltre, si potrebbero eseguire delle ottimizzazioni anche rispetto agli endpoint che nel server gestiscono le richieste dei sensori.

Questi risultati localizzano le DLT sulla curva a S, nella fase di *early adoption*: in questa fase le prestazioni si collocano ancora ad un livello inferiore rispetto alla tecnologia precedente e la fase di crescita rapida non è iniziata.

Non dobbiamo dunque sorprenderci dei risultati emersi dall'analisi delle prestazioni dell'applicazione sviluppata che utilizza IOTA. Anzi, il fatto che si possano identificare alcune peculiarità della tecnologia per cui potrebbe esservi un effettivo vantaggio è un fattore estremamente positivo.

È però possibile a volte che il progredire dei cicli a S nell'evoluzione di una tecnologia venga smorzato senza che venga raggiunto un vero limite tecnologico, ma semplicemente perché le aziende del settore hanno ridotto le loro spese di ricerca e sviluppo. Su questo piano IOTA ha però espresso chiaramente le sue intenzioni sin dal principio, delineando alcune ipotesi che devono essere soddisfatte (specialmente in termini di traffico) affinché la tecnologia possa esprimere il suo pieno potenziale. A questo fine è stata redatta una *timeline* dove regolarmente la fondazione informa la sua community mondiale stabilendo degli obiettivi a breve e lungo termine e aggiornando sullo stato attuale dello sviluppo. Vengono inoltre incoraggiate le iniziative di privati e aziende che intendono testare o implementare il funzionamento della tecnologia, fornendo un supporto costante, un ampio numero di casi "*blueprint*" di esempio e tutte le informazioni possibili sul funzionamento e sui supporti disponibili per lo sviluppo.

Sebbene alcune variabili rappresentino delle incognite e molti aspetti della tecnologia siano ancora acerbi e incompleti, IOTA ha sicuramente posto delle basi solide affinché la sua crescita continui e venga così permesso l'accesso all'interno

[16] per cluster si intende il collegamento di più Docker Host, i quali vengono ospitati sull'infrastruttura fornita da un provider laaS esterno o nel proprio centro di elaborazione dati.

di un nuovo ciclo di crescita della tecnologia, potenzialmente rivoluzionario per molti settori.

Annex

Struttura del codice

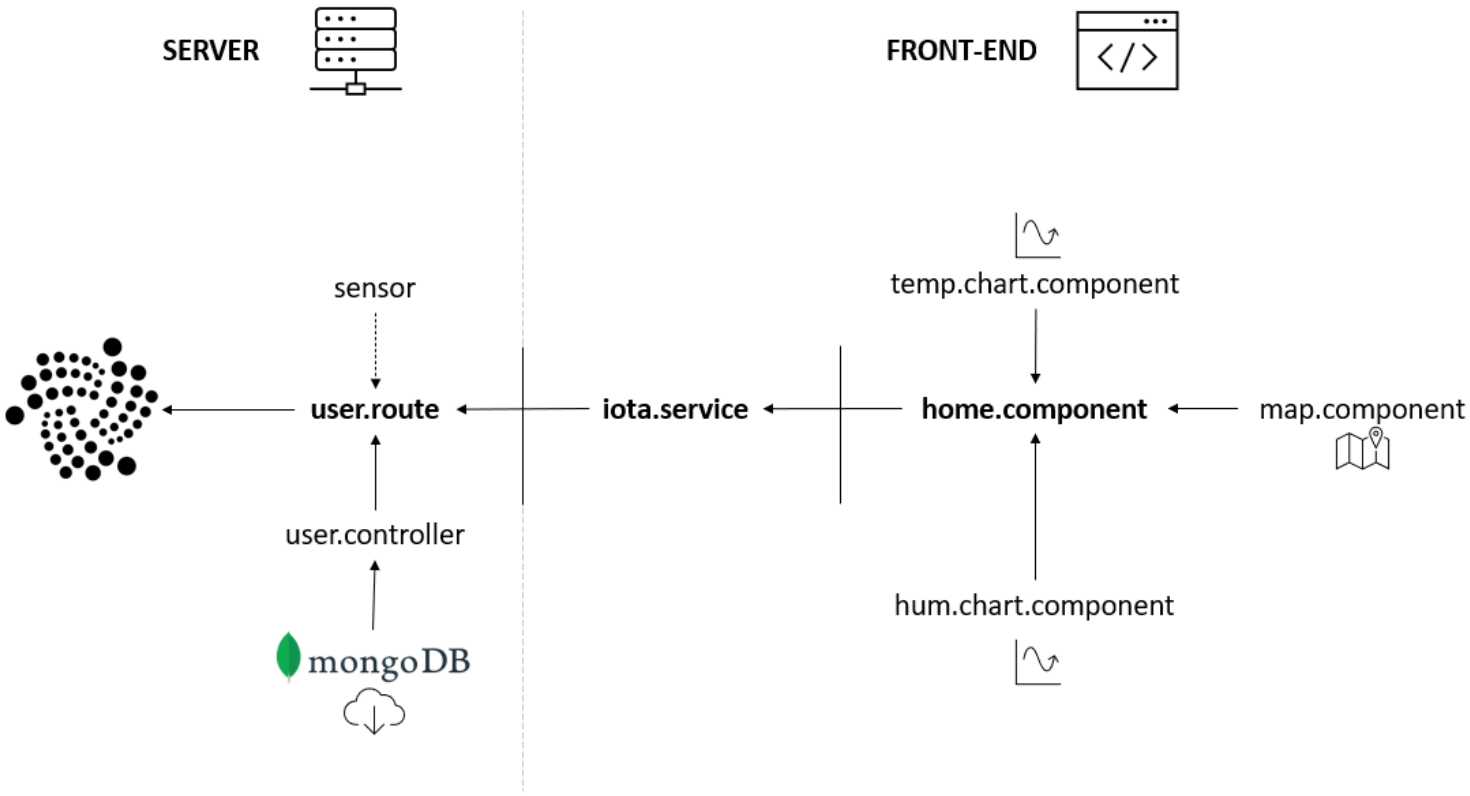


Figura 38-struttura del codice della PoC (app di tracking con IOTA)

Il codice dell'applicazione si può suddividere fondamentalmente in due parti, all'interno delle quali sono situale le singole componenti software.

Il **server** è la parte in NodeJS che comunica direttamente con Iota, tramite lo script `user.route`. Qui sono situati gli endpoint che gestiscono le chiamate provenienti dal front-end e dai sensori, ovvero:

```
router.route('/iota/fetch')
  .get(asyncHandler(fetchIota));
router.route('/iota/example')
  .get(asyncHandler(sendIota));
router.route('/dbexample')
  .get(asyncHandler(searchLast));
router.route('/nodemcu/:id')
  .post(asyncHandler(sendIota));
router.route('/iota/live')
  .get(asyncHandler(fetchIota));
router.route('/iota/findCh')
  .post(asyncHandler(findAllCh))
router.route('/iota/refresh')
  .post(asyncHandler(refresh))
router.route('/iota/scanState')
  .get(asyncHandler(scanMamState))
router.route('/iota/device')
  .get(asyncHandler(showDevice))
router.route('/iota/setChart')
  .get(asyncHandler(chartFetch))
```

Da qui le richieste REST vengono smistate sulle funzioni corrette, dove verrà formulata la risposta. All'interno di queste funzioni lo script interagisce con il web in due modi:

1) Tramite lo script `user.controller` avviene la comunicazione con MongoDB, dove si salvano e si recuperano informazioni relative a: channel ID, MamState e devices (sensori) registrati nella supply chain.
2) Tramite le funzioni di Publish e Fetch vengono invece effettuate interazioni con la rete IOTA, per pubblicare transazioni o recuperare le informazioni richieste dal Client.

Il **front-end** che utilizza Angular è invece esclusivamente eseguito da pc; da qui si possono visualizzare gli spostamenti di magazzino registrati dal sensore, lo storico

delle misurazioni e anche la versione “live” dei dati registrati. Angular tratta le varie componenti della pagina web suddividendole in script differenti (`map.component`, `hum.chart.component`, `temp.chart.component`). Queste sono poi raggruppate in un componente “padre” (`home.component`) e mostrate sulla pagina. Le singole componenti inviano le richieste al server per recuperare le informazioni necessarie; per fare ciò, chiamano le funzioni del file `iota.service`, che gestiscono le chiamate e fanno da “intermediario” tra le due parti.

Di seguito possiamo vedere le funzioni principali di Publish e Fetch, che operano lato server, interagendo con IOTA:

- *Publish*

```
-
-   const publish = async (packet, mamState, postTime) => {
-
-     const trytes = asciiToTrytes(JSON.stringify(packet));
-     const message = Mam.create(mamState, trytes);
-     console.log("\x1b[32m", 'new MAM state after the transaction ha
    s been sent on IOTA tangle', message.state)
-     mamState = message.state;
-     let attach = await Mam.attach(message.payload, message.address,
     3, 10)
-     // console.log("\x1b[32m", 'NEW ATTACH', attach);
-     let timeIOTA = new Date();
-     let publishTime = timeIOTA - postTime;
-     let stringTimeIOTA = timeIOTA.toLocaleString();
-     let stringPostTime = postTime.toLocaleString();
-     console.log('salvo tempistiche', timeIOTA, postTime);
-     console.log('inner publish sensor id', packet.sensorID);
-     await mongoPublishPerformance(stringTimeIOTA, stringPostTime, s
    erverID, publishTime, packet.sensorID);
-
-     console.log('This message have been published on the IOTA MAM C
    hannel:', packet, '\n');
-     console.log("\x1b[35m", 'with root:', message.root, '\n');
-     let toSave = {
-       root: message.root,
-       mamState: mamState
-     }
-     return toSave;
-   }
```

- *Fetch*

```
let fetch = async root => {
      let result = await Mam.fetchSingle(root, currentModeFetch, current
SideKeyFetch);
      console.log('result', JSON.stringify(result));
      if (result.payload != undefined && result.payload != null) {
        totalByte = totalByte + result.payload.length;
        let convertedResult = JSON.parse(trytesToAscii(result.payload));
        console.log('Fetched and parsed', JSON.parse(trytesToAscii(resul
t.payload)));
        dataList.push(convertedResult);
      }
      else {
        let nextRoot = result.nextRoot;
        let dataObj = {
          dataList: dataList,
          nextRoot: nextRoot
        }
        console.log('esco')
        return dataObj
      }
      while (result.payload != undefined && result.payload != null) {
        result = await Mam.fetchSingle(result.nextRoot, currentModeFetch
, currentSideKeyFetch);
        console.log('result', JSON.stringify(result));
        if (result.payload != undefined && result.payload != null) {
          totalByte = totalByte + result.payload.length;
          convertedResult = JSON.parse(trytesToAscii(result.payload));
          console.log('Fetched and parsed', JSON.parse(trytesToAscii(res
ult.payload)));
          dataList.push(convertedResult);
        }
        else {
          let nextRoot = result.nextRoot;
          let dataObj = {
            dataList: dataList,
            nextRoot: nextRoot
          }
          console.log('esco')
          return dataObj
        }
      }
    }
```

Al loro interno sono utilizzate alcune delle API della libreria MAM presentate nel paragrafo 7.3.1.

Settare la rete

- ***Installare il sistema operativo di Raspberry su una scheda SD*** (Raspberry Pi OS, precedentemente chiamato Raspbian). Per fare ciò si possono utilizzare software come balenaEtcher, che eseguono le operazioni in maniera autonoma.

- ***Installare NodeJS e npm sul Raspberry,*** per poter eseguire il server. Ciò si può fare direttamente dal terminale del Raspberry oppure connettendosi da remoto con Git Bash:

  1) Abilitare il repository NodeSource eseguendo il comando seguente nel terminale:

     ```
     curl -sL https://deb.nodesource.com/setup_12.x | sudo bash -
     ```

  2) Una volta abilitato il repository, installare Node.js e npm digitando:

     ```
     sudo apt install nodejs
     ```

- ***Settare il Raspberry come punto di accesso wireless,*** per permettere la ricezione dei dati inviati dal sensore. Le configurazioni del Raspberry dovranno essere impostate con stesso indirizzo IP, ssid e password lungo tutta la supply chain, al fine di poter ricevere le informazioni: nel momento in cui un Raspberry è presente nella rete con caratteristiche differenti, non

sarà in grado di ricevere le informazioni da parte del sensore, le quali andranno quindi perse. Questo processo è abbastanza lungo e consiste di alcuni step:

1) *Installare il software di gestione dei punti di accesso e della rete*

Per poter funzionare come punto di accesso, il Raspberry Pi ha bisogno di avere installato il pacchetto software per punti di accesso hostapd:

```
sudo apt install hostapd
```

Bisogna poi attivare il servizio di "wireless access point" e impostarlo in modo che si avvii di default quando il Raspberry Pi viene acceso:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
```

Per poter fornire servizi di gestione della rete (DNS, DHCP) ai client wireless, il Raspberry Pi deve avere il pacchetto software dnsmasq installato:

```
sudo apt install dnsmasq
```

Infine, installare netfilter-persistent e il suo plugin iptables-persistent. Queste utilty permette di salvare le regole del firewall e ripristinarle quando il Raspberry Pi si avvia:

```
sudo DEBIAN_FRONTEND=noninteractive apt install -y netfilter-
persistent iptables-persistent
```

2) *Definire la configurazione IP dell'interfaccia wireless*

Il Raspberry deve gestire un server DHCP per la rete wireless, al fine di possedere una rete wireless autonoma; questo richiede una configurazione IP statica per l'interfaccia wireless (wlan0). Questo infatti funge anche da router sulla rete wireless, utilizzando il primo indirizzo IP della rete: *192.168.4.1*

Per configurare l'indirizzo IP statico, bisogna modificare il file di configurazione per dhcpcd con:

```
sudo nano /etc/dhcpcd.conf
```

Andare alla fine del file e aggiungere quanto segue:

```
interface wlan0
      static ip_address=192.168.4.1/24
      nohook wpa_supplicant
```

3) *Attivare il routing e il mascheramento IP*

Il Raspberry deve consentire ai client wireless l'acceso a Internet.

Per abilitare l'instradamento, cioè per consentire il flusso di traffico da una rete all'altra tramite il Raspberry, creare un file utilizzando il seguente comando, con il contenuto riportato successivamente:

```
sudo nano /etc/sysctl.d/routed-ap.conf
```

Contenuto del file:

```
#
https://www.raspberrypi.org/documentation/configuration/wire
less/access-point-routed.md
# Abilita il routing IPv4
net.ipv4.ip_forward=1
```

L'abilitazione del routing permetterà agli host della rete 192.168.4.0/24 di raggiungere la LAN e il router principale, accedendo così a internet. Per farlo senza modificare la configurazione del router principale, il Raspberry Pi può sostituire l'indirizzo IP dei client wireless con il proprio indirizzo IP sulla LAN utilizzando la regola del firewall "masquerade".

Il router principale vedrà così tutto il traffico in uscita dai client wireless come proveniente dal Raspberry Pi, permettendo la comunicazione con internet.

Il Raspberry riceverà invece tutto il traffico in entrata, sostituendo gli indirizzi IP e inoltrando il traffico al client wireless che lo richiede.

Questo processo viene configurato aggiungendo un'unica regola firewall:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Salviamo dunque le attuali regole del firewall per IPv4 (inclusa la regola di cui sopra) e IPv6, in maniera tale che vengano caricate all'avvio dal servizio netfilter-persistent:

```
sudo netfilter-persistent save
```

Le regole di filtraggio vengono salvate nella directory `/etc/iptables/`.

4) *Configurare i servizi DHCP e DNS per la rete wireless*

I servizi DHCP e DNS sono forniti da dnsmasq. Il file di configurazione predefinito serve da modello per tutte le possibili opzioni di configurazione, mentre a noi ne servono solo alcune. Dunque, è più facile partire da un file vuoto.

Rinominiamo dunque il file di configurazione predefinito e creiamone uno nuovo:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
```

Aggiungere quanto segue al file e salvarlo:

```
interface=wlan0 # Listening Interface
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
                # Pool of IP addresses served via DHCP
domain=wlan     # Local wireless DNS domain
address=/gw.wlan/192.168.4.1
                # Alias for this router
```

Il Raspberry Pi fornirà dunque indirizzi IP tra 192.168.4.2 e 192.168.4.20, con un tempo di locazione di 24 ore, ai client DHCP wireless. Dovreste essere in grado di raggiungere il Raspberry Pi sotto il nome gw.wlan dai client wireless.

5) *Garantire il funzionamento wireless*

Nel sistema operativo Raspberry Pi OS, la rete wireless a 5 GHz è disabilitata fino a quando l'utente non ha configurato un codice WiFi del paese, di solito come parte del processo di installazione iniziale.

Per garantire che la radio WiFi non sia bloccata sul vostro Raspberry Pi, eseguite il seguente comando:

```
sudo rfkill unblock wlan
```

6) *Configurare il software dell'access point*

Creare il file di configurazione `hostapd`, da situare all'indirizzo `/etc/hostapd/hostapd/hostapd.conf`, per aggiungere i vari parametri della rete wireless.

```
sudo nano /etc/hostapd/hostapd/hostapd.conf
```

Aggiungere le informazioni seguenti al file di configurazione, dove si indicherà l'utilizzo del canale 7, con nome di rete `Iota` e password `iotaiota`.

```
country_code=IT

interface=wlan0

ssid=Iota

hw_mode=g

channel=7

macaddr_acl=0

auth_algs=1
```

```
ignore_broadcast_ssid=0

wpa=2

wpa_passphrase=iotaiota

wpa_key_mgmt=WPA-PSK

wpa_pairwise=TKIP

rsn_pairwise=CCMP
```

Notare la linea country_code=IT: configura il computer per utilizzare le corrette frequenze wireless in Italia. Adattare questa riga e specificare il codice ISO di due lettere del proprio paese.

- ***Installare le librerie MAM e Iota per eseguire il server e inviare le transazioni***

Per fare ciò basta eseguire questo script:

```
npm install @iota/mam @iota/converter @iota/core
```

- ***Scaricare la repository git per poter eseguire il server dell'applicazione***

Per fare ciò sarà sufficiente eseguire il comando git clone seguito dal link del repository di Git dove è situata l'applicazione da linea di comando all'interno della home del Raspberry. Di seguito impostarne l'esecuzione all'avvio, in maniera tale che il server sia subito operativo.

- ***Caricare sulla NodeMcu ESP8266 lo script di Arduino per l'invio dei dati all'un end-poin del server sul Raspberry***.

I dati misurati verranno successivamente pubblicati su IOTA. Come spiegato nel paragrafo 7.2, il *seed* viene inoltrato all'interno della richiesta POST. Tae fattori

rappresenta un punto debole per la sicurezza dell'applicazione, per cui in un caso reale bisognerebbe mascherarne il valore. Lo script è il seguente:

```
#include <DHT.h>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include <ArduinoJson.h> //v5.13.5


const char* ssid = "Iota";

const char* pass = "iotaiota";

IPAddress ip;                    // the IP address of your shield


//Your Domain name with URL path or IP address with path

const               char*               serverName               =
"http://192.168.4.1:4040/api/user/nodemcu/mcu1";

const                 char*                 seed                 =
"VNUVBSAGCBQIAGHAMEROKXSIMTADXVKKFKFNPNVBKACDFMCGLDHQDOKRACYYMHAPQ
GVJJPGIAIMIDQYG9"; //mask it in a real case

#define DHTPIN D2 //pin where the DHT22 is connected

DHT dht(DHTPIN, DHT22);

WiFiClient client;

// the following variables are unsigned longs because the time,
measured in

// milliseconds, will quickly become a bigger number than can be
stored in an int.

unsigned long lastTime = 0;

// Timer set to 10 minutes (600000)

//unsigned long timerDelay = 600000;

// Set timer to 5 seconds (5000)

unsigned long timerDelay = 5000;
```

```
void setup() {

  Serial.begin(9600); // Serial monitor Baudrate

delay(10);

//******************PowerSupply               to               the
Sensor**********************

pinMode(D1, OUTPUT);

pinMode(D3, OUTPUT);

digitalWrite(D1,HIGH);

digitalWrite(D3,LOW);

delay(1000);

//*********************************************************

dht.begin();

}

void loop() {

WiFi.mode(WIFI_OFF);

Serial.println("Trying to Connect with ");

Serial.println(ssid);

WiFi.mode(WIFI_STA);

WiFi.begin(ssid, pass); // Connecting ESP8266 with Internet enabled
Wifi with above mentioned credentials

while (WiFi.status() != WL_CONNECTED)

{

// If the connection was unsuccesfull, it will try again and again

delay(500);

Serial.print(".");

}

// Connection succesfull

Serial.println("");

Serial.println("WiFi connected");

float h = dht.readHumidity(); // Reading Temperature form DHT sensor

float t = dht.readTemperature(); // Reading Humidity form DHT sensor

if (isnan(h) || isnan(t))
```

```
{
Serial.println("Failed to read from DHT sensor!");
return;
}
  //Send an HTTP POST request every 10 minutes
  if ((millis() - lastTime) > timerDelay) {
    //Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED){
      ip = WiFi.localIP();
      Serial.println(ip);
      HTTPClient http;

      // Your Domain name with URL path or IP address with path
      http.begin(serverName);

      // Specify content-type header
      //  http.addHeader("Content-Type",  "application/x-www-form-
urlencoded");
//  Data  to  send  with  HTTP  POST//  String  httpRequestData  =
"api_key=tPmAT5Ab3j7F9&sensor=BME280&value1=24.25&value2=49.54&val
ue3=1005.14";
      // Send HTTP POST request

// int httpResponseCode = http.POST(httpRequestData);

      //  If  you  need  an  HTTP  request  with  a  content  type:
application/json, use the following:
      http.addHeader("Content-Type", "application/json");
      String postStr = "{\"temperatura\":\"";
      postStr += String(t);
      postStr += "\",\"umidita'\":\"";
      postStr += String(h);
      postStr += "\",\"seed\":\"";
```

```
      postStr += String(seed);
      postStr += "\"}";
      Serial.print("body request: ");
        StaticJsonBuffer<300> jsonBuffer;
      JsonObject &root = jsonBuffer.createObject();
        root["temperature"] = t;
        root["humidity"] = h;
        root["seed"] = seed;
        root.printTo(Serial);
      Serial.println(postStr);
      char json_str[300];
        root.prettyPrintTo(json_str, sizeof(json_str));
        int httpResponseCode = http.POST(json_str);
      //int httpResponseCode = http.POST("");
      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      Serial.println(postStr);
      http.end();

 }
    else {
      Serial.println("WiFi Disconnected");
    }
    lastTime = millis();
    delay(20000);
    ESP.restart();
  }
}
```

Completati questi passaggi, l'applicazione della PoC per il tracking di una supply chain con Iota sarà pronta per essere utilizzata

Linea di comando

Nella pagina precedente è mostrato il comando utilizzato per avviare l'applicazione (`npm run start`); una volta che sia il front-end che il back-end saranno stati correttamente eseguiti, sarà possibile ricevere le transazioni dai sensori. Di seguito vediamo cosa succede dal momento in cui viene ricevuta una transazione in poi:

```
C:\Users\facel\myNewProject\IOTA Front-End\iota-application-workspace>npm run start

> mean@2.0.2 start C:\Users\facel\myNewProject\IOTA Front-End\iota-application-workspace
> concurrently -c "yellow.bold,green.bold" -n "SERVER,BUILD" "nodemon server" "ng serve"

[SERVER] [nodemon] 1.19.4
[SERVER] [nodemon] to restart at any time, enter `rs`
[SERVER] [nodemon] watching dir(s): *.*
[SERVER] [nodemon] watching extensions: js,mjs,json
[SERVER] [nodemon] starting `node server`
[SERVER] ../../dist/
[SERVER] server started on port 4040 (development)
[BUILD] i ℹwdsℹ: Project is running at http://localhost:4200/webpack-dev-server/
[BUILD] i ℹwdsℹ: webpack output is served from /
[BUILD] i ℹwdsℹ: 404s will fallback to //index.html
[BUILD]
[BUILD] chunk {main} main.js, main.js.map (main) 563 kB [initial] [rendered]
[BUILD] chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 275 kB [initial] [rendered]
[BUILD] chunk {polyfills-es5} polyfills-es5.js, polyfills-es5.js.map (polyfills-es5) 610 kB [initial] [rendered]
[BUILD] chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
[BUILD] chunk {styles} styles.js, styles.js.map (styles) 575 kB [initial] [rendered]
[BUILD] chunk {vendor} vendor.js, vendor.js.map (vendor) 15.6 MB [initial] [rendered]
Date: 2020-10-14T09:03:27.123Z - Hash: 0f1ae47caa8f751887ce - Time: 32196ms
[BUILD] ** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
[BUILD] i ℹwdmℹ: Compiled successfully.
```

Figura 39-avvio applicazione da liena di comando con "npm run start"

```
 Message received from mcu2 at 2020-10-14T11:46:15.366Z

Current seed PUEOTSEITFEVEWCWBTSIZM9NKRGJEIMXTULBACGFRQK9IMGICLBKW9TTEVSDQMGWKBXPVCBMMCXWMNPDX
Current temperature 27.83
Current humidity 34.60


Mam state before sending new message  {"temperature":"27.83","humidity":"34.60"}
Channel state Initialized
and mode has been set to: restricted

 State correspondence found on MongoDB
 MAM State correctly passed from MongoDB
result [
  {
    _id: 5eea01354497221d30206338,
    subscribed: [],
    channel: {
      side_key: 'SECRETKEY9999999999999999999999999999999999999999999999999999999999999999999999999',
      mode: 'restricted',
      next_root: 'KVGLEGXMZBZSZLKWWWKBCHPKMBIIL9EEWSMBPOKGOJOBZIVPNBYYOXWVP9JBGEWRCRRTSWNQFIDCZJAHJ',
      security: 2,
      start: 2843,
      count: 1,
      next_count: 1,
      index: 0
    },
    seed: 'PUEOTSEITFEVEWCWBTSIZM9NKRGJEIMXTULBACGFRQK9IMGICLBKW9TTEVSDQMGWKBXPVCBMMCXWMNPDX',
    id: 'mcu2',
    serverID: 'node_1',
    timestamp: 2020-06-17T11:40:37.145Z,
    __v: 0,
    current_root: 'OPWPVICSRBAFRBMOLBGDKQDB9OESGXXHTXXZLHWIOTHGWAUIQWUKAE9MGSBXGWLPRCYZUXEWHVCNPACXU'
  }
]
```

Figura 40-responso della publish (parte 1)

- Il messaggio viene ricevuto e vengono mostrati ora, ID del sensore che lo ha inviato, seed del sensore e valori misurati.
- Viene inizializzato un Mam State con il seed ricevuto.
- Viene effettuata la ricerca di un Mam State precedente, relativo al canale del sensore stesso.

```
 SEED COMPARISON

 from MongoDB: PUEOTSEITFEVEWCWBTSIZM9NKRGJEIMXTULBACGFRQK9IMGICLBKW9TTEVSDQMGWKBXPVCBMMCXWMNPDX
from the current NodeMCU PUEOTSEITFEVEWCWBTSIZM9NKRGJEIMXTULBACGFRQK9IMGICLBKW9TTEVSDQMGWKBXPVCBMMCXWMNPDX

 They are the same!
 CHANNEL OPENED!

 new MAM state after the transaction has been sent on IOTA tangle {
  subscribed: [],
  channel: {
    side_key: 'SECRETKEY9999999999999999999999999999999999999999999999999999999999999999999999999',
    mode: 'restricted',
    next_root: 'JFUBBCYZFBCWUDYTNHRGEKOUIPHXOUGS9UCHCQNVBXCBDULMKLPAXBSZI9JVIPX9HLUMYUAFLW9MLMLNR',
    security: 2,
    start: 2844,
    count: 1,
    next_count: 1,
    index: 0
  },
  seed: 'PUEOTSEITFEVEWCWBTSIZM9NKRGJEIMXTULBACGFRQK9IMGICLBKW9TTEVSDQMGWKBXPVCBMMCXWMNPDX'
}
 Channel data correctly passed from MongoDB
salvo tempistiche 2020-10-14T11:46:17.510Z 2020-10-14T11:46:15.366Z
inner publish sensor id mcu2
 Channel data correctly passed from MongoDB
This message have been published on the IOTA MAM Channel: {
  serverID: 'node_1',
  message: '{"temperature":"27.83","humidity":"34.60"}',
  sensorID: 'mcu2',
  timestamp: '2020-10-14 13:46:15'
}

 with root: KVGLEGXMZBZSZLKWWKBCHPKMBIIL9EEWSMBPOKGOJOBZIVPNBYYOXWVP9JBGEWRCRRTSWNQFIDCZJAHJ

post time 2020-10-14T11:46:15.366Z
```

Figura 41-responso della publish (parte 2)

- Se viene trovata una corrispondenza, il Mam State viene aggiornato all'ultima versione salvata e il risultato viene mostrato.

- Viene eseguita l'operazione di publish.

- Vengono mostrate la *root* e l'istante di pubblicazione su Iota della transazione.

```
This mamState will be saved: {
  subscribed: [],
  channel: {
    side_key: 'SECRETKEY9999999999999999999999999999999999999999999999999999999999999999999999999',
    mode: 'restricted',
    next_root: 'JFUBBCYZFBCWUDYTNHRGEKOUIPHXOUGS9UCHCQNVBXCBDULMKLPAXBSZI9JVIPX9HLUMYUAFLW9MLMLNR',
    security: 2,
    start: 2844,
    count: 1,
    next_count: 1,
    index: 0
  },
  seed: 'PUEOTSEITFEVEWCWBTSIZM9NKRGJEIMXTULBACGFRQK9IMGICLBKW9TTEVSDQMGWKBXPVCBMMCXWMNPDX'
}

Number of channels saved on Mongo 4

Searching for a correspondence between channel and sensor:

Trying to publish from sensor mcu2
Sensor correspondence check on MongoDB: mcu1

Trying to publish from sensor mcu2
Sensor correspondence check on MongoDB: mcu2
 sensor state found! mcu2
 MAM State updated on MongoDB collection: mamState
```

Figura 42-responso della publish (parte 3)

- Se viene trovata una corrispondenza, il Mam State viene aggiornato all'ultima versione salvata e il risultato viene mostrato.

Alfabeto tryte – caratteri

Come abbiamo detto, IOTA utilizza il sistema ternario bilanciato. Per rendere i tryte più leggibili all'uomo, il team di sviluppo dello IOTA ha creato l'alfabeto dei tryte: 9ABCDEFGHIJKLMNOPQRSTUVWXYZ. Tale alfabeto viene utilizzato da Iota per tradurre il seed. L'alfabeto tryte è composto da 26 lettere dell'alfabeto latino più il numero 9, per un totale di 27 caratteri. Poiché 1 tryte ha 3^3= 27 combinazioni, ogni tryte può essere rappresentato da un carattere dell'alfabeto tryte.

Alfabeto Tryte		
Tryte	Corrispondente decimale	Carattere
0,0,0	0	9
1,0,0	1	A
-1,1,0	2	B
0,1,0	3	C
1,1,0	4	D
-1,-1-,1	5	E
0,-1,1	6	F
1,-1,1	7	G
-1,0,1	8	H
0,0,1	9	I
1,0,1	10	J
-1,1,1	11	K
0,1,1	12	L
1,1,1	13	M
-1,-1,-1	-13	N
0,-1,-1	-12	O
1,-1,-1	-11	P
-1,0,-1	-10	Q
0,0,-1	-9	R
1,0,-1	-8	S
-1,1,-1	-7	T
0,1,-1	-6	U
1,1,-1	-5	V
-1,-1,0	-4	W
0,-1,0	-3	X
1,-1,0	-2	Y
-1,0,0	-1	Z

Tabella 5-traduzione dei tryte in caratteri

I seed IOTA, gli indirizzi, gli hash, ecc. sono tryte che sono rappresentati da caratteri dell'alfabeto tryte.

Indice delle Figure

Figura 1-struttura "a rete" delle supply chain odierne 7
Figura 2 - differenza tra architettura client/server e P2P 15
Figura 3-valutazione necessità di una blockchain 20
Figura 4-step per l'implementazione di una soluzione blockchain/IoT 23
Figura 5-fork nella blockchain 31
Figura 6-rappresentazione della "doppia spesa" 34
Figura 7-relazione tra potenza computazionale e difficoltà del mining in una rete blockchain 35
Figura 8-classificzaione prodotti "food" in base a quanto sono adatti al tracking con blockchain (secondo Accenture) 48
Figura 9-processo di selezione e implementazione di una soluzione con DLT 56
Figura 10-struttura del Tangle di Iota 59
Figura 11-tips del Tangle 59
Figura 12-altezza dal tip 60
Figura 13-profondità dalla genesi 60
Figura 14-porzione offline di Tangle 61
Figura 15-il collo di bottiglia della blockchian, causato dall'approvazione del blocco 62
Figura 16-Iota è maggiormente scalabile rispetto ad una blockchain 63
Figura 17-motivo per cui bisognerebbe evitare di riutilizzare un indirizzo già utilizzato per inviare denaro 66
Figura 18-percorso pesato del walker dalla genesi al tip 71
Figura 19-caso in cui il walker si sposti solo su transazioni di peso massimo 72
Figura 20-doppia spesa su Iota 73
Figura 21-transazioni confermate e non confermate 74
Figura 22-rappresentazione grafica del Merkle Tree 82
Figura 23-hashing delle leafs e creazione della root 83
Figura 24-variazione di indice e count 84
Figura 25-sezioni di un pacchetto MAM 85
Figura 26-siblings del Merkle Tree 87
Figura 27-creazione della firma del messaggio 88
Figura 28-schema dell'architettura utilizzata nella PoC 97
Figura 29-mappa dove vengono visualizzati gli spostamenti dei lotti tracciati 101
Figura 30-form per esaminare il canale 101
Figura 31-tabella in cui vengono riportate le transazioni presenti sul canale di cui si è eseguita la ricerca 102
Figura 32-funzionalità dei grafici 103
Figura 33-grafici dove sono riportati i valori di temperatura misurati 'live' 103

Figura 34 - diagramma di sequenza dell'operazione di Publish.....109
Figura 35 - diagramma di sequenza dell'operazione di Fetch.....111
Figura 36-processo lineare di innovazione 120
Figura 37-processo ciclico di innovazione di una tecnologia (curve ad S) 121
Figura 38-struttura del codice della PoC (app di tracking con IOTA) 126
Figura 39-avvio applicazione da liena di comando con "npm run start" 141
Figura 40-responso della publish (parte 1) 142
Figura 41-responso della publish (parte 2) 143
Figura 42-responso della publish (parte 3) 144

Indice dei Grafici

Grafico 1-rapporto tra esposizione al rischio e resilienza nelle supply chain di ieri, oggi e domani.....2
Grafico 2-publish time dei due server Raspberry.....113
Grafico 3-distribuzione dei Publish Time sui due server 114
Grafico 4-publish Time in relazione alla temperatura della CPU (1 e 2 sensori attivi).....115
Grafico 5-publish time in relazione alla CPU (molteplici sensori attivi in contemporanea) 116
Grafico 6-andamento del tempo di Fetch 118

Indice delle Tabelle

Tabella 1 - contributo di blockchain e IoT all'interno di una Supply Chain 19
Tabella 2-prestazioni degli algoritmi di consenso 51
Tabella 3-principali caratteristiche di alcuni framework DLT 54
Tabella 4-Performance della Publish al variare dell'intervallo tra le richieste . 118
Tabella 5-traduzione dei tryte in caratteri 145

Bibliografia

Accenture. (2018). *BLOCKCHAIN IN LOGISTICS - Perspectives on the upcoming impact of blockchain technology and use cases for the logistics industry.* Accenture, DHL Trend Research.

Accenture. (2018). *TRACING THE SUPPLY CHAIN - How blockchain can enable traceability in the food industry.* Accenture.

Alan Harrison, R. v. (2008). *Logistics Management and Strategy - Competing through the supply chain.* Prentice Hall - Financial Times.

Bellini, M. (2018, Dicembre 27). *Che cosa sono e come funzionano le Blockchain Distributed Ledgers Technology – DLT*. Tratto da blockchain4innovation.it: https://www.blockchain4innovation.it/esperti/cosa-funzionano-le-blockchain-distributed-ledgers-technology-dlt/

CWIpedia. (2016, Marzo 30). *Container: cosa sono e come funzionano (virtualizzazione)*. Tratto da cwi.it: https://www.cwi.it/data-center/virtualizzazione/container-cosa-funzionano-virtualizzazione-86516

Fastweb. (2017, settembre 29). *Cos'è e come funziona il peer-to-peer*. Tratto da www.fastweb.it: https://www.fastweb.it/internet/cosa-e-come-funziona-p2p/

Frankenfield, J. (2020, Giugno 2020). *Proof of Work* . Tratto da Investopedia: https://www.investopedia.com/terms/p/proof-work.asp

Impact of Industry 4.0 on Supply Chains . (2019, Marzo 08). Tratto da www.gep.com: https://www.gep.com/blog/strategy/impact-of-industry-4-on-supply-chain

Iota. (2017-2020). *Iota Developer Documentation*. Tratto da Iota: https://docs.iota.org/

Iota. (s.d.). *Coordicide*. Tratto da Iota Coordicide: https://coordicide.iota.org/

Laphou Lao, Z. L. (2020). *A Survey of IoT Applications in Blockchain Systems: Architecture, Consensus, and Traffic Modeling.* ACM Computing Surveys.

Mehrdad Salimitari, M. C. (2019). *A Survey on Consensus Protocols in Blockchain for IoT Networks.* Department of Computer Science University of Central Florida.

Michele Nati, J. M.-N. (2019). *Iota Blueprint - Track and Trace PoC.* Iota.

Mobilefish. (2017). *Iota Transaction & Bundle*. Tratto da Mobilefish.com: https://www.mobilefish.com/download/iota/transaction_part10.pdf

Mobilefish. (2017). *Masked Authenticated Messaging.* Mobilefish.com.

Mobilefish. (2017). *Merkle Tree* . Tratto da Mobilefish.com: https://www.mobilefish.com/download/iota/merkle_tree_part18.pdf

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.* Bitcoin.

Olaf Schatteman, D. W. (2020, Aprile 27). *Supply Chain Lessons from Covid-19: Time to Refocus on Resilience*. Tratto da Bain.com: https://www.bain.com/insights/supply-chain-lessons-from-covid-19/

Oracle. (s.d.). *What Is the Internet of Things (IoT)?* Tratto da Oracle.com: https://www.oracle.com/it/internet-of-things/what-is-iot.html

Osservatori.net. (2020, Febbraio 5). Tratto da Come funzionano i Token Blockchain e come si costruiscono: https://blog.osservatori.net/it_it/token-blockchain-come-funzionano

Popov, S. (2018). *The Tangle.* Iota.

Proof-of-Work, spiegata semplicemente. (2018, Febbraio 2018). Tratto da Cointelegraph: https://it.cointelegraph.com/explained/proof-of-work-explained

Team, B. (2018, Settembre 6). *Before Blockchain, There Was Distributed Ledger Technology*. Tratto da medium.com: https://medium.com/blockstreethq/before-blockchain-there-was-distributed-ledger-technology-319d0295f011

UCL Centre for Blockchain Technologies. (2019). *Distributed Ledger Technology in the Supply Chain.* UCL.

Value, I. I. (2017). *Three ways blockchain Explorers chart a new direction.* IBM.

Zia Yusuf, A. B. (2018). *Pairing Blockchain with IoT to cut Supply Chain costs.* Bcg, Cisco.