ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
SEDE DI CESENA

Scuola di Ingegneria ed Architettura
Corso di Laurea in Ingegneria Elettronica e Telecomunicazioni per l'Energia

INTEGRAZIONE TRA BLOCKCHAIN E INTERNET OF THINGS: IMPLEMENTAZIONE, SVILUPPO E ANALISI

Elaborato in
Reti wireless per l'Internet of Things

Tesi di Laurea di:
MARCO GIACHIN

Relatore:
Prof. Ing.
ENRICO PAOLINI
Correlatore:
Chiar.mo Prof. Ing.
MARCO CHIANI

SESSIONE III
ANNO ACCADEMICO 2019–2020

PAROLE CHIAVE

IOT

BLOCKCHAIN

BCOT

CERTIFICATION

EMBEDDED SYSTEMS

Indice

Capitolo 1

Introduzione

Questo elaborato di tesi vuole analizzare, sia con considerazioni di carattere teorico che mediante l'illustrazione di una serie di prototipi realizzati per lo scopo, la possibilitá di integrare la tecnologia *blockchain* con dispositivi *Internet of Things (IoT)*, realizzando il paradigma definito *Blockchain of Things (BCoT)*.

Negli ultimi anni la tecnologia *blockchain* si é distinta sia per i risultati portati nel campo dello scambio di valore digitale, tramite le *criptovalute*, sia per la possibilitá di utilizzarla come alternativa ai tradizionali strumenti di memorizzazione delle informazioni. Una peculiaritá della tecnologia *blockchain* infatti é quella di conservare le informazioni in "blocchi", collegati gli uni agli altri tramite tecniche crittografiche; il risultato di questa particolare struttura dati é la possibilitá di tracciare tutta la storia di un dato e i suoi scambi, arrivando al punto in cui é stato inserito all'interno del sistema.
Per garantire la realizzabilitá e la validitá di questo tipo di processo viene abbandonato l'uso, per conservare l'informazione e recuperarla al bisogno, di sistemi centralizzati, controllati da singole aziende o gruppi di queste, come ad esempio i sistemi cloud di proprietá di Google, Aruba o simili. Con la tecnologia *blockchain* viene per la prima volta definito un ecosistema del tutto decentralizzato, privo di autoritá centrale che controlli gli scambi informativi e che possa quindi teoricamente modificarne il contenuto o nascondere parte di esse.

Contemporaneamente allo sviluppo della tecnologia *blockchain* l'IoT ha trovato campo di applicazione in una quantitá enorme di applicazioni industriali, al fine di effettuare il monitoraggio e il controllo da remoto o automatico di sistemi elettronici. I dispositivi IoT sono fondamentali per lo sviluppo delle cosiddette *smart cities*, poiché abilitano l'automazione di ti-

pologie di azioni e controlli che normalmente dovevano essere realizzati in maniera manuale. La peculiaritá dell'*IoT* sta nell'intuizione di collegare ad internet direttamente i dispositivi che raccolgono dati o che effettuano operazioni di controllo; questi dispositivi sono solitamente *embedded*, cioé sistemi on-chip di piccole dimensioni, progettati per svolgere specifiche attivitá tramite un'unica scheda integrata. Nel paradigma dell'*IoT* i dispositivi non comunicano con l'essere umano per richiedere comandi o istruzioni ma sono in grado di comunicare tra loro in maniera autonoma ed effettuare operazioni, anche complesse, prendendo decisioni in maniera autonoma, a volte supportati da algoritmi di *Machine Learning* o *Intelligenza Artificale*.

La commistione delle due tecnologie sopracitate ha preso il nome di *Blockchain of Things (BCoT)*, e rappresenta l'ultima evoluzione dello scambio informativo tra dispositivi *IoT*, che sono quindi in grado, oltre che di effettuare operazioni sull'ambiente, anche di certificare i dati raccolti in maniera automatica tramite la *blockchain* e fornirli all'utente finale all'interno di un ledger decentralizzato (letteralmente "libro mastro", termine inglese usato per indicare la struttura dati che sostiene una generica *blockchain*).

Lo spunto per la realizzazione di questo elaborato di tesi é stato la partecipazione ad un progetto Climate KIC "Pathfinder", dal titolo "Introducing blockchain technology in professional WEEE management (BlockWEEE)"; l'obiettivo di quest'ultimo era quello di validare e analizzare la possibilitá di integrazione della tecnologia blockchain all'interno dei modelli di business delle aziende facenti parte della cosiddetta *industria 4.0*. Il progetto, nel quale sono stati coinvolti diversi *stakeholders*, era focalizzato sul tracciamento lungo l'intera supply chain, incluso il fine vita, di apparecchiature elettroniche di tipo professionale. Al termine della loro vita tali prodotti divengono *Waste from Electrical and Electronic Equipmente (WEEE)*: questo acronimo inglese si riferisce ai rifiuti provenienti da apparecchiature elettroniche ed elettriche, le quali hanno una regolamentazione a parte per lo smaltimento e riciclo.

Durante lo sviluppo del progetto e anche successivamente ad esso, nella realizzazione vera e propria dell'elaborato e dei prototipi *IoT*, sono stati analizzati altri campi di applicazione possibili, in quanto ritenuti piú adatti all'applicazione delle tecnologie studiate. Primo tra tutti é stato analizzato il campo dell'*Agrifood*, comprendente tutti i business case che fanno parte della produzione (ed eventuale raccolta), trasporto e vendita di beni alimentari. Come verrá spiegato in seguito, gli attuali modelli di lavoro risultano negli ultimi anni sempre piú inadatti alle richieste del pubblico e la *BCoT*

potrebbe rappresentare un serio candidato per risolvere alcune criticitá. Lo stesso tipo di analisi é stata svolta anche per tutti i campi di applicazione che presentano la necessitá di raccogliere, gestire e infine fornire a utenti terzi una serie di dati piú o meno sensibili, che devono essere certificati e potenzialmente tracciati.

Lo scopo di questo elaborato é quindi quello di proporre una soluzione general-purpose che faccia uso della tecnologia *BCoT* per rispondere ad un insieme di criticitá trasversali e riscontrate in vari campi applicativi. Nell'implementazione finale a cui si vuole tendere la *blockchain* dovrá essere trattata al pari di un "black-box" all'interno del quale convergono informazioni e dal quale queste possano essere prelevate. I dispositivi *IoT* vedono la *blockchain* tramite una *API* senza conoscere la sua struttura interna, per permettere al sistema il massimo livello di riusabilitá e genericitá, con integrazione del tipo "plug-and-play".

Lo studio teorico é stato accompagnato da una serie di incontri e confronti con aziende e realtá che basano su dispositivi *IoT* i loro modelli di lavoro, al fine di intercettarne le criticitá e definire delle soluzioni.

Nella prima parte verranno presentati dal punto di vista teorico la tecnologia *blockchain*, partendo dalla sua prima implementazione con *Bitcoin*, e arrivando al concetto piú ampio delle *Distributed Ledger Technologies (DLT)*; nello specifico si cercherá di delineare una lista di requisiti che questa tecnologia deve rispettare affinché essa sia integrabile con dispositivi *IoT* embedded. Grande attenzione verrá posta su tre tecnologie analizzate e che poi costituiranno la base per i prototipi *BCoT* realizzati, quali: *Hyperledger Fabric*, *BigchainDB* e *Iota*.
Successivamente verrá esposta l'attivitá realizzata nell'ambito del progetto ClimateKIC e i risultati ottenuti, ponendo particolarmente attenzione alla prima implementazione del sistema realizzata, evidenziandone tutte le criticitá e gli sviluppi futuri possibili definiti a fine progetto; a questo seguirá poi la presentazione di un prototipo aggiornato del sistema *BCoT*, realizzato per superare le problematiche del primo sistema. Per concludere verrá presentato un *Proof of Concept (PoC)*, che si intende sviluppare con lavori futuri, il cui obbiettivo ultimo é quello di realizzare una *BCoT* come "black-box".

Capitolo 2

Blockchain e DLT: da Bitcoin a IOTA

A fine 2008 un utente (o un gruppo di utenti), avente come nickname "Satoshi Nakamoto" e la cui vera identitá non é mai stata rivelata, pubblicó su internet un whitepaper dal titolo "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. Con questo documento ebbe inizio l'era delle criptovalute e della tecnologia *blockchain* che, a dodici anni dalla sua introduzione, ha le potenzialitá per guidare la piú grande rivoluzione dello scambio di informazioni dopo la creazione della rete internet stessa.

Svolgendo le ricerche preliminari per la stesura dell'elaborato é emersa quasi subito la necessitá di allargare il campo anche alle *Distributed Ledger Techologies (DLT)*. In linea generale possiamo includere all'interno di questo insieme tutti i sistemi di scambio e condivisione di informazioni che sfruttano un sistema decentralizzato, a prescindere dall'effettiva implementazione del *ledger* che viene scelta; in questo senso, quindi non tutte le *DLT* sono *blockchain* ma tutte le *blockchain* sono *DLT*.

2.1 Bitcoin: la nascita della blockchain

2.1.1 La rivoluzione della moneta peer-to-peer

Il motivo fondamentale per cui é stato pensato il sistema di *Bitcoin* [1] puó essere individuato nel concetto di fiducia. Si pone infatti l'accento sul fatto che i modelli di scambio di valore (e in linea generale di informazione) abbiano la necessitá di autoritá terze di fiducia che siano in grado di intermediare controllando le transazioni; il controllo centralizzato é atto ad impedire quello che viene definito "double-spending", situazione in cui un utente spende due volte lo stesso ammontare di moneta sfruttando l'impossibilitá del beneficiario di controllare personalmente le transazioni non dirette a lui.

Con *Bitcoin* viene proposto il primo sistema di pagamento elettronico basato su prova crittografica, tramite il quale due controparti possono effettuare la transazione, senza necessitá di autoritá terze di fiducia, di monete digitali dette *BTC*. Il sistema viene reso sicuro dalla decentralizzazione dello stesso ed é possibile dimostrare che, finché i nodi onesti, nel loro complesso, possiedono una capacitá computazionale (a volte riferita come *hashing power*) superiore rispetto a quelli malevoli, é impossibile effettuare un "double-spending".

2.1.2 Le transazioni in Bitcoin: l'uso della crittografia e della hashing

Nella terminologia di *Bitcoin* uno scambio di valuta elettronica corrisponde ad una catena di firme digitali che permettono il trasferimento di valore da un proprietario al seguente. Ogni utente possiede una chiave privata, di sua esclusiva proprietá, che gli permette di firmare digitalmente delle informazioni; il fatto che la chiave sia strettamente privata impedisce che chiunque possa firmare informazioni sul *ledger* al posto dell'utente che la possiede: questa svolge sostanzialmente il ruolo di una impronta digitale elettronica che marca l'identitá dell'utente. La chiave privata é generata dal sistema in maniera pseudocasuale, come un numero compreso tra 1 e $n = 1.158 \cdot 1077 - 1$ e puó essere visualizzata in formato esadecimale come una sequenza di 64 caratteri alfanumerici.

Dalla chiave privata k viene generata poi una chiave pubblica K tramite una funzione non invertibile. In termini crittografici una funzione viene definita non invertibile quando, conoscendone il risultato, il valore a partire dal quale il risultato é stato generato puó essere ottenuto solo mediante un

approccio a forza bruta, computazionalmente molto oneroso. Nel caso di Bitcoin K viene ricavato mediante una funzione non invertibile denominata *elliptic curve point multiplication*, e nello specifico viene implementata una *Elliptic Curve Digital Signature Algorithm (ECDSA)*. Il risultato della funzione é un punto su una curva ellittica in un campo Z_p, generato a partire da k, usando la curva denominata *secp256k1* [2], la cui equazione é la seguente:

$$y^2 mod(p) = (x^3 + 7) mod(p) \tag{2.1}$$

dove p é un numero primo di valore:

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \tag{2.2}$$

In Figura 2.1 viene riportato un esempio della forma della curva se lavorassimo con numeri reali; é tuttavia da considerare che, essendo nel caso di *Bitcoin* calcolata nel campo Z_p, l'aspetto sarebbe quello di una costellazione di punti sparsi.

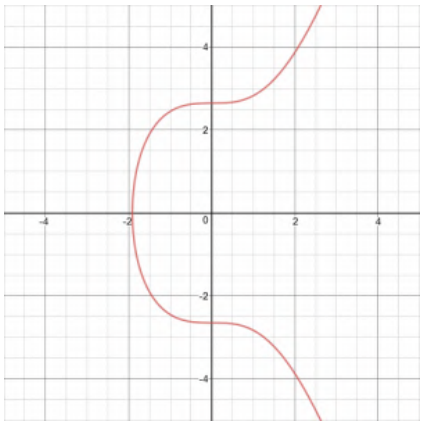


Figura 2.1: Curva secp256k1 calcolata su numeri reali ($y^2 = x^3 + 7$) [2]

Data la chiave privata k calcoliamo quindi K come $K = G \cdot k$ dove G é il punto generatore. La chiave pubblica corrisponderá quindi ad un punto sulla curva ellittica di coordinate $K = (x, y)$, dove anche x e y sono rappresentabili in esadecimale da 64 caratteri. Essendo la funzione non invertibile l'unico modo per calcolare k, anche conoscendo K e G, é procedere per tentativi; questo risulta in una complessitá computazionale non trattabile per la dimensione dei numeri coinvolti.

Per scambiare valuta il proprietario P_0 quindi firma digitalmente una hash della transazione precedente T_{n-1} (quella che ha generato la valuta o che l'ha

fornita a P_0 tramite uno scambio) insieme alla chiave pubblica K_1 del beneficiato, cioé P_1. La transazione T_n conterrá quindi la hash cosí costruita e sará collegata a P_1 in quanto vi é apposta la sua chiave pubblica $k1$. Qualora P_1 voglia scambiare a sua volta questa valuta realizzerá una hash di T_n, insieme alla chiave pubblica del nuovo beneficiario P_2, e la firmerá digitalmente, realizzando la transazione T_{n+1}, e cosí via realizzando una catena di blocchi (Figura 2.2).

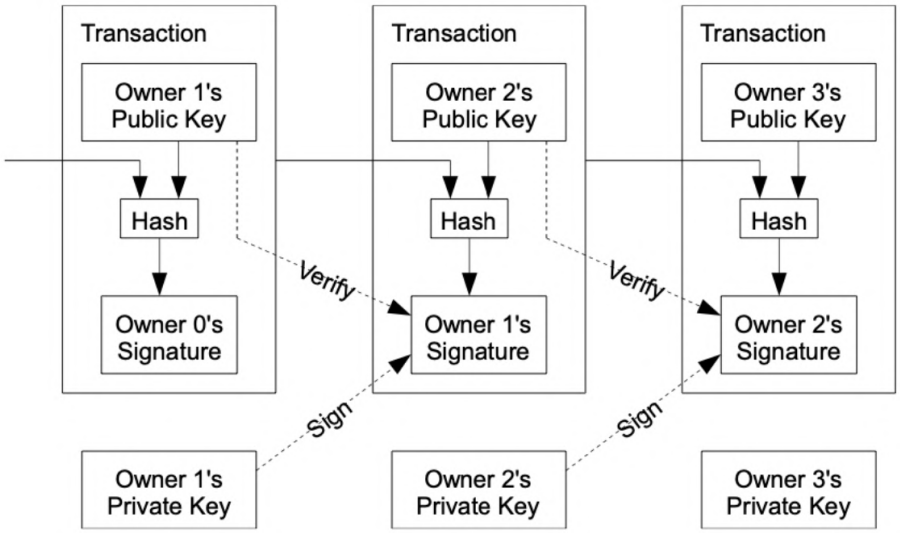


Figura 2.2: Rappresentazione schematica della catena di blocchi realizzata da tre utenti [1]

2.1.3 Risoluzione del problema dell'autoritá centrale: server marcatura temporale

Il problema di questa struttura é che risulta ancora impossibile per l'utente P_1 verificare che la quantitá di valuta scambiata con T_n non sia stata giá scambiata da P_0 con una transazione precedente o parallela (il cosiddetto "double-spending"), poiché non gli é possibile ispezionare tutte le transazioni fatte da P_0. Nei sistemi tradizionali questo viene risolto deputando ad un'autoritá terza, detta "zecca", il controllo delle transazioni, assumendo che P_0 non scambi direttamente valuta con P_1 ma la passi alla zecca, la quale ne emette una nuova di uguale valore e poi la trasferisce a P_1; questo sistema funziona fintanto che si assume che la moneta emessa dalla zecca non sia mai stata spesa due volte, il che concentra il potere di gestione dell'infrastruttura, e quindi la sua affidabilitá, interamente nelle mani della societá che gestisce

la zecca.

La decentralizzazione si ottiene annunciando pubblicamente tutte le transazioni su un *ledger* condiviso e sostenuto su base peer-to-peer dagli utenti stessi che utilizzano il sistema, sfruttando questi per confermare, tramite un algoritmo di raggiungimento del consenso, il passato delle transazioni e l'ordine in cui queste sono state effettuate. Viene quindi immaginato un server di marcatura temporale distribuito, in grado di fare hash di un gruppo di oggetti marcandoli temporalmente per poi pubblicare la hash ottenuta (Figura 2.3). La marcatura temporale assicura che i dati devono essere esistiti in una determinata data, in quanto sono finiti all'interno della hash e la marcatura allo step n contiene quella effettuata allo step $n-1$ e cosí via, realizzando una vera e propria *block-chain (catena di blocchi)*.

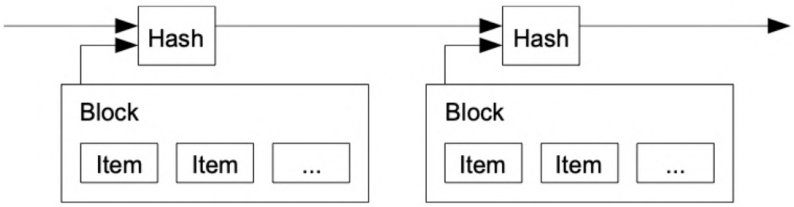


Figura 2.3: Funzionamento della marcatura temporale a catena [1]

2.1.4 Il raggiungimento del consenso: problema dei generali bizantini

Argomento chiave dei sistemi distribuiti in generale é il raggiungimento del consenso, inteso come la necessitá che la maggioranza dei nodi concordi sull'informazione da pubblicare o sull'operazione da effettuare. Il problema puó essere posto tramite il dilemma logico detto "Problema dei generali Bizantini" [3], dal quale prende il nome una caratteristica fondamentale dei sistemi decentralizzati su base crittografica, cioé la *Byzantine Fault Tolerance (BFT)* (Figura 2.4). Il problema consiste nell'ipotizzare una situazione in cui un certo numero di generali (nel paragone con la *blockchain* i nodi del network), posizionati in diverse posizioni sul campo di battaglia, deve prendere una decisione del tipo: attaccare o non attaccare in maniera sincronizzata, in modo da poter agire contemporaneamente; unica specifica del problema é che la decisione deve essere non invertibile una volta presa. La difficoltá sta nel fatto che i generali non possono comunicare tra loro ma solo mediante dei messaggeri (nel nostro caso i messaggi scambiati tra i nodi attraverso la rete

internet) e che questi possono smarrire il messaggio, recapitarlo in ritardo oppure recapitarlo errato. Inoltre un generale potrebbe decidere, per qualsiasi motivo, di agire in maniera disonesta e di non agire secondo la decisione del gruppo o di recapitare un messaggio falso per confondere gli altri.

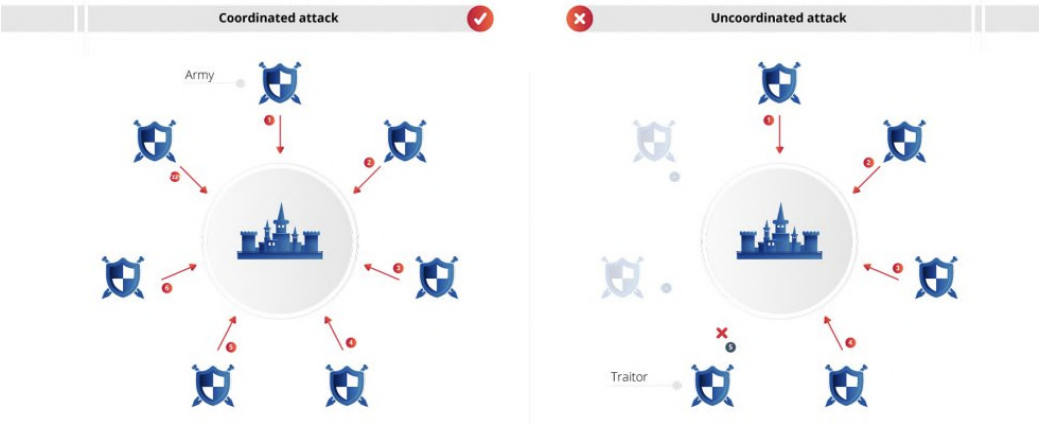


Figura 2.4: Rappresentazione grafica del Problema dei Generali Bizantini

Nel contesto della blockchain abbiamo quindi un certo numero di nodi del sistema decentralizzato che comunicano tra loro scambiandosi le informazioni riguardo alle transazioni da controllare al fine di convergere ad una decisione del tipo: transazione valida (e quindi nessun *double-spending*) o transazione non valida. Uno dei nodi potrebbe peró in qualsiasi momento decidere di inserire una transazione non valida per proprio tornaconto personale, oppure semplicemente un messaggio potrebbe venire perso o alterato dalla rete.

La BFT é quindi la proprietá di un sistema di resistere ad una serie di errori riconducibili al Problema dei Generali Bizantini. Per ottenere questa caratteristica esistono differenti metodi, detti *algoritmi di consenso*, con i quali é possibile assicurare che il sistema converga alla soluzione corretta anche quando un certo numero di errori intenzionali o no sono presenti.

2.1.5 Algoritmi di consenso: Proof-of-Work

Per *Bitcoin* viene proposto un algoritmo di consenso del tipo *Proof-of-work (PoW)*: questa tipologia di algoritmi prevede che ogni utente debba spendere potenza computazionale per risolvere un “puzzle crittografico” di difficoltá

variabile in base alle specifiche del sistema.

L'algoritmo viene applicato a "blocchi" facenti parte della *blockchain*, ognuno dei quali contiene le transazioni effettuate tra il calcolo della hash precedente e quello corrente (Figura 2.5). Il *PoW* di *Bitcoin* viene implementato tramite un double *SHA-256*[5], dove viene quindi calcolata la hash *SHA-256* di una hash *SHA-256* [6]. L' algoritmo *PoW* implementato prevede, come incentivo per i partecipanti ad agire in maniera onesta, la necessitá di spendere molta potenza computazionale per effettuare il calcolo, il quale consiste nel modificare la nounce del blocco T_n, la quale viene posposta alla hash del blocco T_{n-1}, finché la hash di T_n non comincia con un predeterminato numero di zeri. Il lavoro medio richiesto alla CPU é proporzionale al numero di zeri richiesti e questo cambia in base allo sviluppo della tecnologia, per assicurare un numero sempre costante di hash calcolati all'ora, che é all'incirca mantenuto pari a sei.

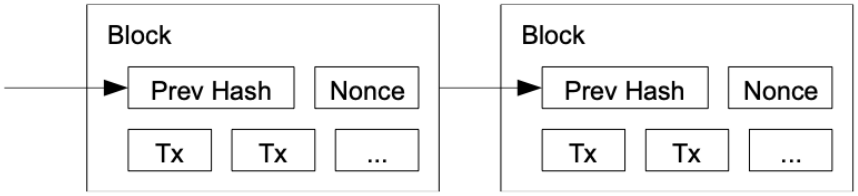


Figura 2.5: Concatenazione dei blocchi a seguito dell'esecuzione del PoW

L'introduzione di questa operazione crittografica impedisce sostanzialmente di modificare il contenuto di un blocco senza dover eseguire di nuovo l'intero lavoro di cifratura; oltre a questo, la potenza del sistema sta nel fatto che per modificare una transazione di indice n, effettuata prima rispetto a quella attuale di indice z, l'utente dovrebbe eseguire nuovamente il lavoro di cifratura per tutte le $z-n$ transazioni successive ad n, in quanto queste sono concatenate dopo n e la hash attuale di $n+1$ é ottenuta con quella originale di n.

Una volta risolto il puzzle la soluzione, che corrisponde al prossimo blocco da inserire nella catena, viene inviata in broadcast a tutti gli altri nodi, che la accettano nel momento in cui ricominciano a risolvere il puzzle partendo dal blocco appena ricevuto, allungando quindi la catena. La maggioranza viene quindi espressa dalla catena piú lunga che viene creata, cioé quella su cui é stato speso piú sforzo di *PoW* e che quindi verrá maggiormente estesa con blocchi successivi.

L'algoritmo di consenso usato da *Bitcoin* lascia tuttavia spazio a situazioni ambigue che potrebbero essere potenzialmente fallimentari nel caso in cui non venissero gestite in maniera corretta. La logica della *blockchain* prevede l'estensione di una catena con il nuovo blocco nel momento in cui gli utenti hanno raggiunto il consenso sulla transazione e controllato che le hash siano state calcolate in maniera corretta e non vi sia *double-spending*. Puó accadere tuttavia che due utenti risolvano il puzzle nello stesso momento, dando vita a due blocchi, entrambi validi, che vengono inviati entrambi a tutti gli altri nodi (Figura 2.6). Per vari motivi alcuni nodi potrebbero ricevere T_B e altri T_{B2} come blocco successivo a T_A e originare quindi una "fork" nella catena.

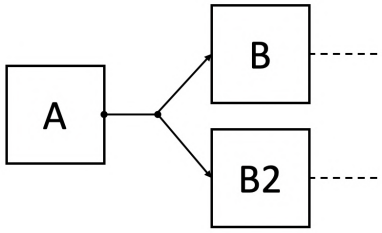


Figura 2.6: Situazione appena dopo la fork

La risposta del sistema a questa situazione é detta "Longest Chain Rule", regola secondo la quale il sistema mantiene sempre la catena piú lunga, e quindi quella su cui é stata spesa piú potenza di *PoW*. Supponendo che ci siano piú utenti che lavorano sul puzzle a partire da T_B é probabile (non sicuro, in quanto la risoluzione del puzzle é del tutto casuale) che uno di questi riesca a risolverlo prima di un utente che lavora su T_{B2}, estendendo quindi la catena superiore. Anche se é possibile anche lo scenario inverso, su lunghi periodi possiamo supporre che ad essere estesa sia sempre la catena su cui lavorano piú nodi. Una volta risolto il puzzle per il blocco T_B, la catena inferiore, in quanto piú corta, viene abbandonata e tutti i blocchi ritornano a lavorare su T_C (Figura 2.7).

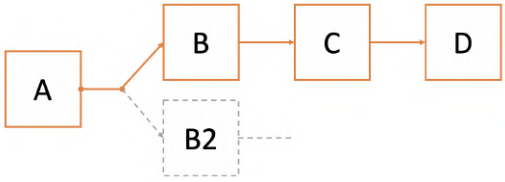


Figura 2.7: Risultato dell'applicazione della longest chain rule su una fork: la catena grigia viene abbandonata

L'algoritmo *PoW* di *Bitcoin* é quindi utilizzabile per raggiungere il consenso fintanto che la maggioranza della potenza di calcolo é detenuta da nodi onesti, i quali alla lunga permetteranno alla catena onesta di crescere piú in fretta e superare catene concorrenti o create da utenti malintenzionati. La probabilitá che un utente malintenzionato raggiunga il lavoro dei nodi onesti diminuisce infatti in maniera esponenziale mano a mano che vengono aggiunti blocchi; in termini generali si puó rapportare la probabilitá che l'utente ha di estendere la catena alla sua percentuale di "hash-rate" (cioé la frazione dell'effettiva potenza computazionale totale che é posseduta dall'utente).

Come ulteriore incentivo all'utente che riesce a generare, in gergo "minare", un blocco, é concessa la possibilitá di aggiungere una transazione di valore pari a qualche BTC diretta a se stesso; viene quindi generata nuova valuta secondo un meccanismo periodico. Se consideriamo che un blocco viene generato ogni dieci minuti sappiamo che 525600 blocchi sono generati ogni anno; senza regolamentare il numero di BTC assegnati per ogni blocco il numero di monete in commercio aumenterebbe costantemente. Per questo motivo *Bitcoin* effettua il cosiddetto "halving", cioé il dimezzamento della ricompensa ogni quattro anni, che é quindi passata da 50 BTC nel 2008 a 25 nel 2012 e attualmente corrisponde a 6.25. Questo meccanismo assicura che il numero massimo di BTC generati sia limitato e che il valore in linea del tutto teorica cresca nel tempo, essendo la valuta sempre piú rara. Poiché un utente riceve la sua ricompensa quando il blocco da lui proposto viene accettato all'interno della "longest chain"é controproducente in termini probabilistici per un utente malevolo spendere risorse per realizzare un blocco falso, in quanto questo ha scarse probabilitá di entrare a far parte della catena valida.
Possiamo considerare due scenari, di complessitá crescente, per dimostrare l'effettiva inefficacia del comportamento scorretto:

- Un utente malevolo cerca di cambiare una transazione che é stata spesa n blocchi fa sulla catena piú lunga;
- Un utente effettua una transazione e la cambia non appena il beneficiario l'ha accettata.

Nello specifico possiamo considerare il primo scenario come una "gara" tra utente onesto P_O e utente malintenzionato P_M nell'estendere la catena di blocchi. A livello statistico possiamo modellare la situazione come una *Binomial Random Walk (BRW)*, dove l'evento "successo", cioé l'estensione della catena onesta, aumenta il vantaggio di P_O come $+1$, mentre l'evento

"fallimento" estende di un blocco la catena di P_M, riducendo il divario tra i due di un fattore -1. Il valore di interesse é la probabilitá q_z che l'utente malevolo riesca a raggiungere il pareggio tramite un numero infinito di tentativi, partendo da una situazione nella quale P_M si trova in svantaggio di z blocchi. Questo é modellabile come il problema di "Gambler's Ruin", nel quale abbiamo:

- p = probabilitá che l'utente onesto riesca a generare il prossimo blocco;
- q = probabilitá che l'utente malevolo riesca a generare il prossimo blocco;
- q_z = probabilitá che l'utente malevolo riesca a recuperare partendo z blocchi di svantaggio.

Risulta:

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases} \tag{2.3}$$

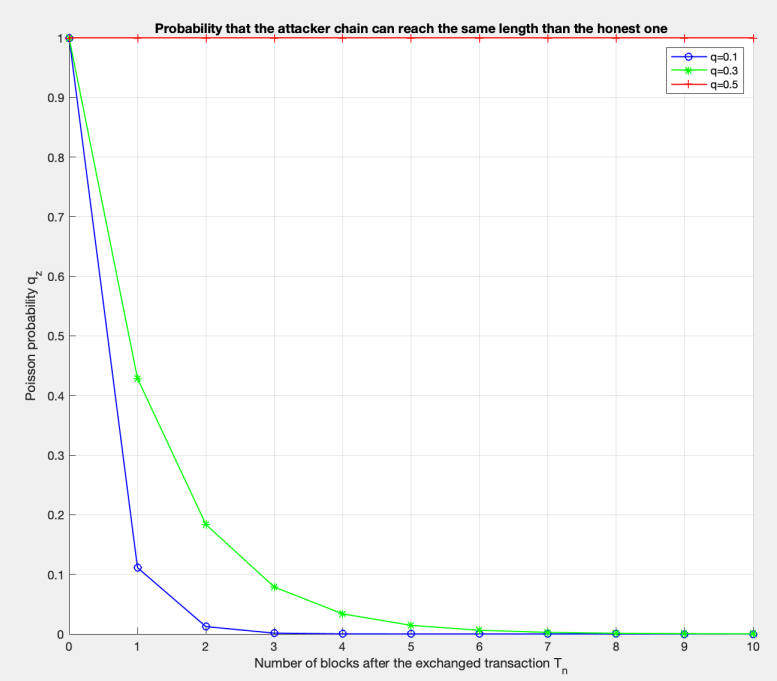


Figura 2.8: Probabilitá di raggiungere il pareggio in una *BRW* al variare di q

Data l'ipotesi di base che la potenza di calcolo dell'utente malevolo sia inferiore a quella dell'utente onesto e che quindi $p > q$ allora la probabilitá q_z decresce esponenzialmente mano a mano che aumenta il gap di blocchi da raggiungere (Figura 2.8). A meno di non avere particolare fortuna nel momento in cui la catena é stata estesa di un solo blocco le probailitá diventano subito estremamente piccole. Come si puó notare dalla simulazione, l'uso di valori per q superiori a 0.5 fornisce risultati per q_z inammissibili e fa cadere anche l'ipotesi di uso del "Gambler's Ruin".

Il secondo problema posto é invece piú complesso da modellare e prevede che un utente malevolo voglia scambiare valuta con un beneficiario e poi cambiare la transazione per effettuare un "double-spending". Per evitare ogni possibilitá che P_M possa preparasi in anticipo una catena di blocchi P_O gli fornisce la chiave pubblica da usare per lo scambio appena prima e, una volta che la transazione é inviata, P_O aspetta che venga estesa da un numero z di blocchi prima di considerarla valida. Parallelamente P_M, dopo avere effettuato la transazione, inizia a costruire una catena parallela con la versione alternativa della sua transazione. Una volta che la catena principale é stata estesa da z blocchi non é possibile conoscere con esattezza quale sia il progresso fatto da P_M ma, supponendo che per un blocco malevolo si impieghi lo stesso tempo che per un blocco onesto, il progresso é dato da una distribuzione di Poisson con parametro:

$$\lambda = z \cdot \frac{q}{p} \tag{2.4}$$

Possiamo quindi calcolare la probabilitá che l'utente P_M possa recuperare dal punto raggiunto, dove ha calcolato k blocchi, con $k < z$, quando risulta ancora in svantaggio di $z - k$ blocchi:

$$P_z = \sum_{k=0}^{\infty} \frac{\lambda^k \cdot e^{-\lambda}}{k!} \cdot \begin{cases} (\frac{q}{p})^{(z-k)} & \text{if } k \leq z \\ 1 & \text{if } k > z \end{cases} \tag{2.5}$$

In Figura 2.9 é riportato il risultato per valori di q crescenti e anche in questo caso la probabilitá decresce esponenzialmente con l'aumento di z, anche se piú lentamente rispetto al caso precedente, in quanto P_M non parte da zero ma da k blocchi giá minati.

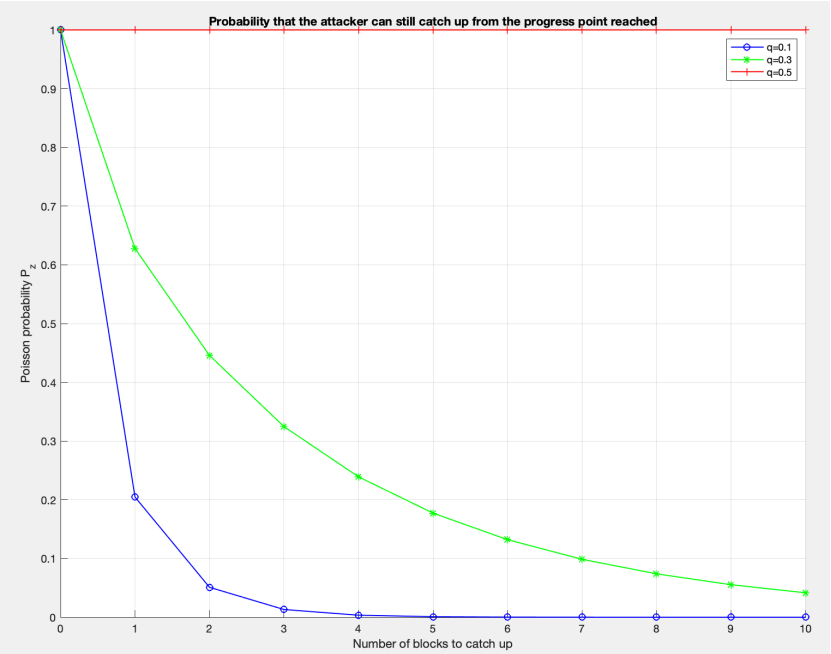


Figura 2.9: Probabilitá, al variare di z, che l'utente malintenzionato possa recuperare dopo aver minato k blocchi

2.1.6 Considerazioni finali su Bitcoin

Le caratteristiche di *Bitcoin* e della sua *blockchain* lo rendono perfetto come caso studio e hanno permesso il suo mantenimento come prima valuta digitale in termini di interesse e capitalizzazione di mercato negli ultimi anni.

Tuttavia, il protocollo di *Bitcoin*, per quanto sia un punto di partenza per tutte le future implementazioni di *blockchain*, risulta ad oggi profondamente inefficiente sotto svariati punti di vista:

- Consumo eccessivo di potenza per la realizzazione del PoW;
- Bassa *transaction rate*, pari a un blocco ogni dieci minuti;
- Alto costo dell'hardware necessario per effettuare il "mining", il ché porta alla necessitá di delegare a terzi il compito di inserire transazioni per gli utenti comuni, aggiungendo una serie di commissioni;
- Scarsa possibilitá di personalizzare il protocollo o il contenuto dei blocchi.

Basti pensare che, al momento di realizzazione di questo elaborato, il consumo di *Bitcoin* é stimato in $75TWh$ all'anno, il che la rende decisamente la *blockchain* meno adatta ad ogni tipologia di progetto che si ponga il problema ambientale.

Per tutti questi motivi non si ritiene che quella di *Bitcoin* possa essere la *blockchain* adatta a sviluppare un sistema di scambio di informazioni e dati come é invece obiettivo di questo elaborato. Inoltre, la registrazione di transazioni sulla *blockchain* di *Bitcoin* ha costi non trascurabili e non é effettuabile direttamente da hardware a basso costo; questo ne rende ulteriormente sconveniente l'uso quando la necessitá é l'integrazione con dispositivi IoT, che fanno del risparmio energetico e del basso costo il loro punto chiave.

Questo protocollo é stato quindi analizzato allo scopo di illustrare le caratteristiche principali delle *blockchain* "tradizionali", di tipo pubblico e *permissionless*, al fine di comprendere meglio le caratteristiche che verranno ricercate in seguito ed effettuare un confronto tra gli strumenti tradizionali e le loro evoluzioni, con i relativi pro e contro.

2.2 Evoluzione delle blockchain da pubbliche a private: il caso Hyperledger

La scelta della tecnologia *blockchain* tradizionale come strumento di scambio dell'informazione ci permette di garantire sistemi robusti, trasparenti, stabili e affidabili. Questo é garantito anche dal principio di replicazione, secondo il quale ogni nodo che partecipa al network mantiene una copia completa della transazioni e puó essere interrogato da chiunque in maniera del tutto libera. Al contrario dei database (DB) tradizionali, quindi, il fuori servizio di un singolo nodo o gruppo di questi non é significativo fintanto che la maggioranza dei nodi continua a funzionare correttamente. Il sistema é stato inoltre dimostrato essere essenzialmente stabile in quanto garantisce immutabilitá alle informazioni, le quali sono protette dalle modifiche anche da parte dello stesso utente che le ha inserite originariamente.

Queste caratteristiche si dimostrano peró a volte incompatibili con alcuni modelli di business aziendali, che tuttavia ricercano l'uso della tecnologia *blockchain* per i vantaggi che questa é comunque in grado di fornire. É nata cosí l'idea di sviluppare *blockchain* nelle quali la caratteristiche di decentralizzazione viene eliminata parzialmente o completamente, fornendo soltanto ad un gruppo scelto di utenti la possibilitá di interagire con il sistema.

2.2.1 Blockchain pubbliche, private e permissioned

In via generale, possiamo classificare le *blockchain* come pubbliche o private. Sebbene questa classificazione non sia ancora del tutto accettata in ambito prettamente scientifico, questo elaborato si pone l'obiettivo di presentare soluzioni che possano essere utilizzate all'interno dei modelli di business aziendali, e pertanto le considerazioni fatte si baseranno sui feedback ricevuti da questo tipo di ambiente.

Una *blockchain* tradizionale é di principio pubblica, nel senso che ogni utente é in grado di effettuarvi transazioni, unendosi al network, e potenzialmente anche di validare le transazioni effettuate da altri partecipando attivamente al raggiungimento del consenso. Oltre a questo é sempre consentita, senza limiti di accesso, la lettura dei dati presenti sul *ledger* da parte di ogni utente che ne abbia necessitá. La completa trasparenza del sistema é garantita e sostenuta dai nodi facenti parte della rete e costituisce un fattore importante per la creazione della fiducia condivisa necessaria a mantenere funzionante il protocollo. Questa trasparenza viene poi compensata dal fatto che il *ledger* mostra in chiaro solo le chiave pubbliche degli utenti ma non fornisce alcuna informazione riguardo all'identitá reale del possessore di una chiave.
Le *blockchain* pubbliche maggiormente utilizzate ad oggi risultano sicuramente *Bitcoin* ed *Ethereum* e sono utilizzate principalmente per scambiare valuta digitale, anche se quest'ultima presenta ulteriori caratteristiche che verranno accennate brevemente in seguito. La seconda detiene ad oggi il primato come sistema *blockchain* piú utilizzato per applicazioni industriali, grazie alla presenza degli *smart contracts*, programmi in esecuzione sulla *blockchain* stessa, in grado di gestire condizioni complesse ed effettuare automaticamente transazioni. La possibilitá di effettuare transazioni viene sorretta dalla presenza di grandi aziende, detti *Exchange*, che permettono agli utenti che non siano *miners* di acquistare valuta e scambiarla sia con altri utenti che con la piattaforma stessa.

A fianco alle *blockchain* pubbliche gli ultimi anni hanno visto la nascita delle *blockchain* private, sistemi sempre distribuiti ma i cui nodi sono detenuti da una serie di utenti autorizzati dall'autoritá centrale che ha richiesto la costruzione del network. In questo caso quindi non abbiamo un sistema decentralizzato e auto-sostenuto da una comunitá indipendente ma un vero e proprio strumento di condivisione di dati privati condiviso tra utenti selezionati e sfruttato per fornire vantaggi all'interno di determinati modelli di business. Una implementazione valida per una *blockchain* privata é quella in

cui un'azienda decida di voler condividere informazioni con tutti gli attori di una catena di produzione o con le aziende amiche o partner. In questo caso una *blockchain* privata puó essere realizzata distribuendo i nodi in maniera selettiva e garantendo l'accesso a questi solo a predeterminati utenti.

Le *blockchain* pubbliche e private hanno quindi essenzialmente lo stesso funzionamento ma differiscono nel grado di decentralizzazione che raggiungono e nella tipologia di accesso che viene fornito ai partecipanti al network. Mentre nelle prime la decentralizzazione é totale e il livello di accesso uguale per tutti gli utenti, nella seconde é l'autoritá che ha disposto la realizzazione del network a stabilire i permessi di accesso, la quantitá di nodi presenti e gli utenti che possono inserirvi informazioni; per questo motivo le *blockchain* pubbliche vengono a volte dette "permissionless" e quelle private "permissioned".

Le *blockchain* private utilizzano spesso algoritmi di consenso diversi e meno complessi rispetto a quelli usati dalle *blockchain* pubbliche, in quanto é anche statisticamente inferiore il rischio di avere utenti malevoli all'interno del network ed é comunque presente un'autoritá centrale che possa gestire gli utenti. Un caso particolarmente rilevante di *blockchain* privata é rappresentato da *Hyperledger Fabric*.

Le *Blockchain* private condividono comunque con quelle pubbliche una serie di caratteristiche fondamentali che ne motivano l'interesse per applicazioni industriali. Primo tra tutti é il fatto che in entrambi i *ledger* i dati vengono inseriti sotto forma di nuovi blocchi di una catena e non possono quindi essere eliminati o modificati in seguito senza che il sistema se ne accorga automaticamente e risolva l'errore. In secondo luogo inoltre entrambe le implementazioni garantiscono la replicazione dell'informazione su tutti i nodi, garantendo quindi servizi peer-to-peer *fault tolerant* e relativamente stabili. É inoltre da sottolineare come in entrambe le implementazioni il mantenimento del sistema sia dipendente dalla capacitá dei nodi di raggiungere e mantenere il consenso sul corretto contenuto informativo dei vari blocchi.

Dalla loro parte le *blockchain* private hanno generalmente il fatto che, essendo presenti meno nodi all'interno del network ed essendo gli algoritmi di consenso piú semplici, le transazioni sono piú veloci da eseguire e quindi é possibile aumentare il *transaction rate*, in termini di *transactions per seconds (TPS)*, oltre che rendere il sistema piú facilmente scalabile. Questo viene tuttavia compensato da una maggiore difficoltá nel generare fiducia in maniera decentralizzata e in una maggiore vulnerabilitá in caso di utenti malintenzionati all'interno del network.

2.2.2 Un framework per blockchain private: Hyperledger

Hyperledger [7] é un framework realizzato dalla collaborazione tra la *Linux Foundation* e la comunitá internazionale, atto a fornire un sistema completamente open-source per costruire soluzioni basate su *blockchain* che siano inseribili in modelli di business avanzati.

Il principio di uso di *Hyperledger* é analogo a quello di un qualsiasi framework open-source, quindi all'utente che voglia interfacciarvisi é fornita una documentazione base e una serie di componenti installabili sul proprio PC ed estensibili a piacere, con i quali realizzare la propria rete.

Hyperledger Fabric é una delle varie implementazioni possibili di *Hyperledger* e permette la realizzazione di *blockchain* permissioned dove i nodi vengono gestiti dal proprietario della rete, tipicamente un'azienda singola o un gruppo di aziende, il quale é l'unico con l'autoritá per permettere a questi di instaurare una comunicazione tra loro e partecipare al network. I motivi per i quali si é ritenuto *Fabric* un valido strumento per lo sviluppo di questo progetto saranno chiariti piú avanti, nel *Capitolo 3*.
Le parti che contribuiscono alle transazioni non saranno poi solo normali utenti ma altre organizzazioni o aziende, le quali possono possedere uno o piú nodi forniti dall'organizzazione proprietaria del network; quello che si realizza cosí é un consorzio dove le aziende hanno la possibilitá di gestire e configurare i nodi ad esse forniti.
I nodi possono poi anche essere configurati per permettere ad utenti esterni generici di visualizzare le informazioni o parte di esse, tramite un livello di accesso personalizzato, in modo che queste possano essere comunicate anche all'esterno del network in maniera sicura; in questo caso parliamo di *end nodes*.

La particolaritá di *Fabric* sta nel permettere la creazione di canali distinti all'interno dello stesso nodo, fornendo quindi la possibilitá effettiva di garantire differenti livelli di accesso e permessi ad utenti differenti riguardo a informazioni diverse; i canali cosí creati sono realizzabili per essere del tutto separati anche se presenti sullo stesso nodo, come se stessimo effettivamente realizzando due catene di blocchi parallele che vengono estese in maniera del tutto indipendente l'una dall'altra. É inoltre possibile, esattamente come in *Ethereum*, definire degli *Smart Contracts* che stabiliscano regole di funzionamento del sistema; con *Smart Contract* si indica essenzialmente un programma scritto per funzionare direttamente sull'infrastruttura *blockchain*

e che é in grado di interagire con gli utenti e con le informazioni in un gran numero di modalitá. La facilitá offerta dai canali, unita a quella degli *Smart Contracts*, garantisce la possibilitá di abilitare una grande varietá di modelli di business tramite *Fabric*; test interni di *IBM* hanno dimostrato come sia possibile mantenere contemporaneamente fino a 25 canali sullo stesso nodo. Nell'implementazione di *Fabric* tramite canali ogni utente possiede un certificato con il quale si presenta ai nodi al momento della richiesta di effettuare una transazione; all'interno del certificato sono elencati i suoi permessi e questo viene confrontato con lo *Smart Contract*, in questo caso detto *Chaincode*, per garantire che l'operazione richiesta con la transazione sia permessa all'utente in questione.

In *Fabric* i nodi possono essere di tre tipi:

1. Endorser: hanno il compito di ricevere le richieste di transazioni dalle applicazioni fornite ai vari client. Una volta ricevuta la richiesta validano la transazione (controllando i certificati allegati), eseguono il *Chaincode* e simulano il risultato della transazione, senza peró modificare il ledger, per capire se questa debba essere approvata o rifiutata;

2. Anchor: sono i nodi responsabili di ricevere gli aggiornamenti e comunicarli agli altri nodi facenti parte dello stesso canale. Questo tipo di nodi puó essere "scoperto" da parte degli altri nodi senza l'intervento diretto dell'organizzazione che ha configurato la rete;

3. Orderer: sono i nodi che si occupano della comunicazione all'interno del network. Questi mantengono e modificano lo stato del *ledger*, creando i blocchi e inviandoli in broadcast a tutti i peer. L'implementazione piú comune per questi avviene tramite *Kafka*, un software di scambio messaggi con alta "throughput fault tolerance".

2.2.3 Flusso delle transazioni e raggiungimento del consenso in Hyperledger Fabric

Il flusso di processo (Figura 2.10) di una generica transazione su *Fabric* é il seguente: un partecipante di una organizzazione invia una richiesta di transazione al proprio nodo *endorser*, il quale esegue il *Chaincode* e simula la transazione, rispondendo al client con il responso "transazione accettata" o "transazione rifiutata"; a questo punto se la transazione é stata accettata il client la invia ai nodi *orderer* che la includono in un blocco e la inviano in broadcast a tutti i nodi *anchor* delle differenti organizzazioni della rete, i quali poi la ritrasmetteranno ai vari peer interni per sincronizzare i *ledger*.

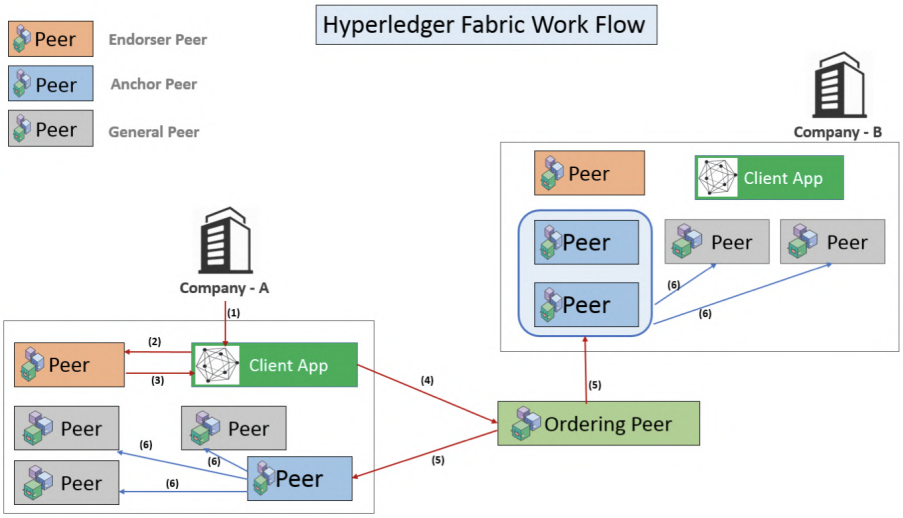


Figura 2.10: Schema di un generico ecosistema Hyperledger con due organizzazioni e due gruppi di nodi [8]

Per raggiungere il consenso tra i vari nodi *orderer* viene usato un algoritmo elettivo chiamato *Raft* [9], il quale si basa su una strategia ad elezione, dove in fase di configurazione i nodi convergono alla decisione di un "master" che riceverá le future transazioni e le ritrasmetterá in broadcast a tutti gli altri; le specifiche tecniche [10] dell'algoritmo non verranno approfondite in quanto non sono di interesse ai fini dell'elaborato.

2.2.4 Modalitá di conservazione dei dati in Fabric

L'implementazione di *Fabric* permette di conservare due tipologie di dati:

- On-chain;
- Off-chain.

Mentre i primi sono dati riguardanti le transazioni e sono salvati in blocchi con un identificativo unico e una hash, i secondi sono documenti di qualsiasi tipo, foto, video, file pdf o in formato testuale; questa caratteristica permette di usare *Fabric* per scambiare e conservare come in un vero e proprio database anche i documenti che devono essere scambiati tra utenti appartenenti a diverse organizzazioni all'interno dello stesso consorzio.

Hyperledger, nella sua implementazione *Fabric*, non consente la possibilitá di effettuare delle fork sulla catena, obbligando quindi tutti i nodi a mantenere sempre la stessa copia delle informazioni e garantendone quindi la completa consistenza.

É altresí possibile utilizzare *Fabric* per realizzare *blockchain* pubbliche, creando un singolo certificato per un utente generico, che possa essere riutilizzato da tutti per proporre transazioni alla rete e che passi la verifica del *Chaincode*; vista la presenza dei canali é anche possibile far coesistere sullo stesso nodo una *blockchain* privata e una pubblica.

2.2.5 Considerazioni finali su Hyperledger Fabric

Al contrario di *Bitcoin Fabric* espone una serie di funzionalitá che lo rendono compatibile con un utilizzo a livello industriale, e sono giá presenti alcuni casi di implementazioni della tecnologia in campo reale. La sua enorme complessitá di utilizzo lo rende peró uno strumento oggi ancora di difficile adozione, sia per la difficoltá pratica di utilizzo e la non completezza della documentazione, sia per la mancanza di un numero adeguato di casi studio che possano validare in anticipo la possibilitá di risolvere un effettivo bisogno industriale.

La caratteristica di versatiliá di *Hyperledger* e la sua varietá di implementazioni sono tuttavia un grande incentivo al suo sviluppo e alla sua adozione poiché permettono un livello di personalizzazione avanzato e la possibilitá di adattare la rete alle piú disparate esigenze. Inoltre, il fatto di poter realizzare canali privati insieme a canali pubblici, rende possibile soddisfare la richiesta, da parte delle aziende, di riservatezza riguardo a dati sensibili ad uso prettamente interno; contemporaneamente é peró possibile soddisfare, almeno in parte, la richiesta del pubblico di visualizzare le informazioni in maniera trasparente, eliminando il paradigma classico per il quale é necessario fidarsi di chi fornisce le informazioni, il quale tuttavia molto spesso é direttamente interessato a manipolarle od ometterne una parte.

Come verrá mostrato piú avanti, *Fabric* si é rivelato un ottimo strumento per la realizzazione della *BCoT*, ma é stato abbandonato in un secondo momento a causa della sua elevata complessitá, che lo rende effettivamente applicabile in maniera efficiente soltanto a casi d'uso dove la rete da realizzare sia talmente complessa da motivare lo sforzo: per esempio qualora siano presenti numerose organizzazioni e utenti, oltre che un certo numero di canali. Per implementazioni di complessitá inferiore, come quelle analizzate ai

fini di questo elaborato, sono state preferite tecnologie diverse, che verranno illustrate nelle prossime sezioni.

2.3 Verso i blockchain database: la rivoluzione di bigchainDB

BigchainDB é un progetto open-source annunciato nel Febbraio 2016 e proposto inizialmente in una prima versione 0.1 che presentava caratteristiche di tipo "alpha" piuttosto che quelle di un prodotto "ready-to-market". Il design originale consisteva in un DB tradizionale con l'aggiunta di alcune caratteristiche tipiche della blockchain, come decentralizzazione, immutabilitá e asset "owner-controlled"; con quest'ultimo termine si intende la possibilitá di rappresentare digitalmente oggetti reali tramite la *blockchain* e assegnarne la proprietá in maniera univoca.

Nella versione attuale, la 2.0 [11], sono state aggiunte funzionalitá atte a rendere il sistema *BFT* e ad eliminare un certo livello di centralizzazione ancora presente a causa della struttura del DB utilizzato, che consisteva in un master node, deputato ad effettuare le registrazioni delle informazioni, affiancato ad altri nodi che semplicemente ne replicavano le operazioni.

2.3.1 Come bigchainDB riesce a garantire le caratteristiche di DB e blockchain

Un DB tradizionale, sia esso *SQL* o di altro tipo, fornisce agli utenti la possibilitá di conservarvi all'interno dati, ma anche di rimuoverli o modificarli a piacimento, purché gli utenti possiedano i privilegi di amministratore sul sistema. Questa tipologia di sistemi é solitamente utilizzata per le elevate prestazioni in termini di transazioni al secondo che permette di raggiungere, oltre che per la bassa latenza nell'inserire informazioni; é inoltre garantita agli utenti la possibilitá di ricercare informazioni all'interno della struttura dati, indicizzandole secondo i campi che contengono, tramite le cosiddette "queries".

Le caratteristiche di base della *blockchain*, quali decentralizzazione, immutabilitá e gestione del consenso non sono presenti nei sistemi database tradizionali. Con lo sviluppo della tecnologia *blockchain* tuttavia é cresciuto sempre di piú l'interesse, anche a livello industriale, di sistemi ibridi che siano in grado di garantire i vantaggi della *blockchain* uniti all'alta versatilitá dei

DB, ad un alto *TPS* e alla bassa latenza. Con *bigchainDB* viene proposto il primo modello ibrido che concentra le caratteristiche di queste due tecnologie all'interno di un unico sistema (Figura 2.11), cercando di realizzare un prodotto "production-ready" che possa essere sostituito ai tradizionali DB all'interno di differenti modelli di business.

	Typical Blockchain	Typical Distributed Database	BigchainDB
Decentralization	✓		✓
Byzantine Fault Tolerance	✓		✓
Immutability	✓		✓
Owner-Controlled Assets	✓		✓
High Transaction Rate		✓	✓
Low Latency		✓	✓
Indexing & Querying of Structured Data		✓	✓

Figura 2.11: Confronto tra database tradizionali, blockchain e bigchainDB 2.0 [11]

BigchainDB sfrutta il software *Tendermint* [12] per la parte di consenso e scambio di messaggi; questo ha un meccanismo di raggiungimento del consenso di tipo elettivo, che garantisce al sistema la caratteristica di essere *BFT*, e viene implementato su tutti i nodi con cui viene realizzato il network *blockchain*. Ogni nodo ha al suo interno un DB *MongoDB* locale, in modo che, se un utente malintenzionato dovesse avere privilegi di amministratore su un singolo nodo, non sarebbe in grado di corrompere l'intera rete, lasciando protetti gli altri DB. Nel momento stesso in cui ogni nodo della rete é detenuto e amministrato da un utente diverso, la rete diventa automaticamente decentralizzata in quanto viene a mancare il "single point of control and failure" tipico dei DB tradizionali. La struttura di una generica rete *bigchainDB* con quattro nodi é riportata in Figura 2.12.

Per quanto riguarda l'immutabilitá, il sistema semplicemente non fornisce alcun tipo di *API* che permetta di modificare o cancellare dati. Inoltre, questa caratteristica é ancora una volta garantita dal fatto che ogni nodo conserva una copia completa dei dati in un DB interno "stand-alone" quindi, anche qualora un nodo venisse corrotto, gli altri manterrebbero comunque

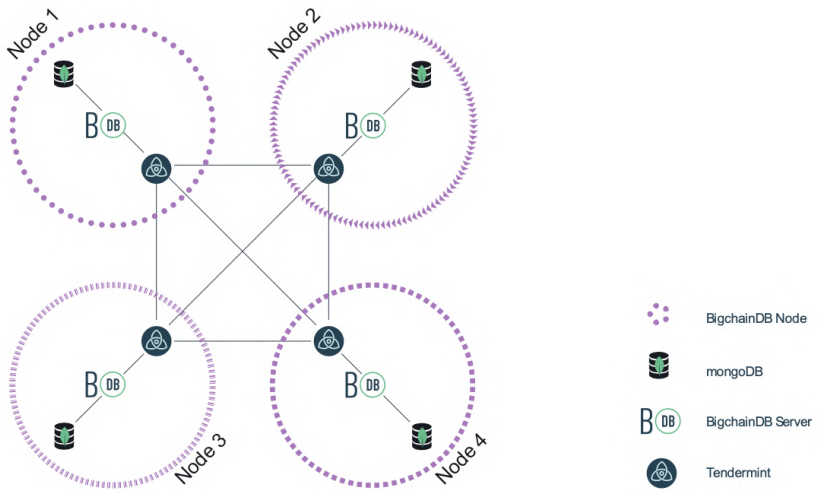


Figura 2.12: Schema di una rete bigchainDB con 4 nodi intercomunicanti [13]

una copia completa e corretta dei dati. A queste due soluzioni si aggiunge l'uso della crittografia, poiché ogni utente in *bigchainDB* é identificato da una coppia chiave pubblica, chiave privata (K, k), la prima ottenibile dalla seconda tramite funzione non invertibile; ogni transazione inserita sul network, e quindi ogni blocco che viene firmato, é immutabile senza cambiarne la firma e quindi senza che questa operazione sia individuabile da parte degli altri utenti.

2.3.2 Le transazioni in bigchainDB

BigchainDB é costruito per realizzare due soli tipi di transazioni:

1. CREATE: transazione che crea un nuovo blocco o asset;

2. TRANSFER: transazione che trasferisce un asset precedentemente creato.

La struttura generale di una transazione su *BigchainDB* (in questo caso una CREATE) é riportata in Figura 2.13:

Ogni transazione presente su *bigchainDB* riporta questa forma generica e viene fornita tramite un oggetto stringa *JSON*:

```
{
  "id": "3667c0e5cbf1fd3398e375dc24f47206cc52d53d771ac68ce14d
  ↪ df0fde806a1c",
  "version": "2.0",
  "inputs": [
    {
      "fulfillment": "pGSAIEGwaKW1LibaZXx7_NZ5-V0alDLvrguGLyL
      ↪ RkgmKWG73gUBJ2Wpnab0Y-4i-kSGFa_VxxYCcctpT8D6s4uTGOO
      ↪ F-hVR2VbbxS35NiDrwUJXYCHSH2IALYUoUZ6529Qbe2g4G",
      "fulfills": null,
      "owners_before": [
        "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "outputs": [
    {
      "amount": "1",
      "condition": {
        "details": {
          "public_key":
          ↪ "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY",
          "type": "ed25519-sha-256"
        },
        "uri": "ni:///sha-256;d-_huQ-eG-QQD-GAJpvrSsy7lLJqyNh
        ↪ tUAs_own7aTY?fpt=ed25519-sha-256&cost=131072"
      },
      "public_keys": [
        "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "operation": "CREATE",
  "asset": {
    "data": {
      "message": "Greetings from Berlin!"
    }
  },
  "metadata": null
}
```


Figura 2.13: Struttura di una transazione su bigchainDB

- id: corrisponde alla hash della transazione e permette di controllare che non sia stata modificata, oltre che di ricercarla all'interno del DB;
- inputs: lista di oggetti composti da chiavi pubbliche e altri campi; per i nostri scopi viene utilizzata per definire il proprietario precedente dell'asset a cui si riferisce la transazione;
- outputs: lista di oggetti composti da chiavi pubbliche e altri campi, tra cui anche delle "condizioni"; per i nostri scopi, essa viene utilizzata per definire il proprietario attuale dell'asset a cui la transazione si riferisce;
- asset: oggetto JSON che riporta l'asset che sta venendo creato o trasferito;
- metadata: oggetto JSON che permette di allegare informazioni aggiuntive alla transazione.

Tramite gli inputs e gli outputs é possibile scambiare asset, o parte di questi, e definire politiche anche relativamente complesse di trasferimento, con una serie di condizioni da inserire all'interno degli output, sotto forma di stringhe, che una transazione TRANSFER deve soddisfare per poter "spendere" un input ricevuto.

Per la creazione delle transazioni sono disponibili in rete alcuni driver, sviluppati per vari linguaggi, che rendono il processo piú semplice.

Grazie alla presenza di inputs e outputs *bigchainDB* offre la possibilitá di realizzare "owner-controlled assets", cioé asset (che possono essere token digitali, valuta, oggetti o qualsiasi altra cosa) che possono essere trasferiti solo dal proprietario; questa caratteristica fa parte del gruppo di quelle ereditate dalla *blockchain*.

La struttura di *bigchainDB* permette agli utenti di effettuare delle transazioni di tipo CREATE, che creano l'asset desiderato; questa transazione, effettuata dall'utente generico P_A, viene firmata con la sua chiave privata k_A e inviata al network. Una volta creato un asset (Figura 2.14) viene quindi inserita una transazione sulla *blockchain*, identificata tramite un ID; la transazione contiene in chiaro la chiave pubblica K_A e solo P_A puó effettuare operazioni su questa. L'altra operazione che puó essere effettuata é quella di TRANSFER, tramite la quale P_A, usando la propria chiave privata k_A, é in grado di trasferire il controllo dell'asset ad un generico utente P_B, utilizzando la chiave pubblica K_B per indicare il nuovo proprietario a cui viene passata la proprietá della transazione e dell'asset ivi contenuto.

La transazione cosí creata conterrá all'interno dell'asset l'ID della transazione precedente nella catena, cioé quella che é stata trasferita a P_B. Qualora poi P_B voglia trasferire di nuovo l'asset potrá realizzare una TRANSFER verso P_C usando K_C e firmando la transazione con k_B, e cosí via. In ognuna di queste operazioni *bigchainDB*, tramite *Tendermint*, controlla comunque che la transazione non stia cercando di effettuare un "double-spending".

2.3.3 Creare un asset e trasferirlo: un esempio pratico

Per mostrare in maniera piú diretta il funzionamento di *bigchainDB*, e per effettuare una serie di test su un progetto a cui si fará cenno nel *Capitolo 5*, si é deciso di implementare un semplice nodo *bigchainDB* tramite una immagine

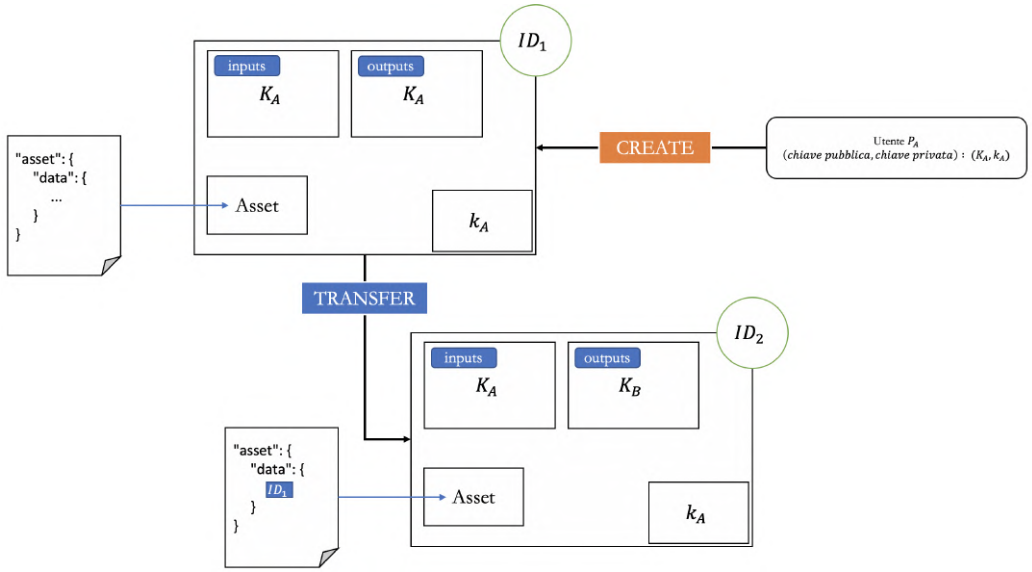


Figura 2.14: Percorso di una transazione che crea un asset e lo trasferisce due volte

Docker su un PC con sistema operativo MacOS. Possiamo quindi configurare in maniera diretta un nodo su un'istanza di *Docker* con il seguente comando:

```
$ docker run \
  --detach \
  --name bigchaindb \
  --publish 9984:9984 \
  --publish 9985:9985 \
  --publish 27017:27017 \
  --publish 26657:26657 \
  --volume $HOME/bigchaindb_docker/mongodb/data/db:/data/db \
  --volume $HOME/bigchaindb_docker/mongodb/data/configdb:/data/configdb \
  --volume $HOME/bigchaindb_docker/tendermint:/tendermint \
  bigchaindb/bigchaindb:all-in-one
```

Una volta che il nodo é operativo espone naturalmente un'API, che comprende le seguenti route:

- `http://localhost:9984/api/v1/outputs?public_key=...` : permette di visualizzare tutte le transazioni presenti con una determinata chiave pubblica;
- `http://localhost:9984/api/v1/transactions/id` : permette di ricercare una transazione sulla base del suo ID.

Sono state quindi generate tre coppie di chiavi pubbliche e private, associate agli utenti P_A,P_B,P_C, le quali parti pubbliche sono:

- K_A = 3LrM7Pv9RrNvqD12RUgYJwQR86hG8apRne9ADGbUy4Ut;
- K_B = GdAxyeikX7B4tf41mQww7dr65KFmeNv54kP3syGvFUwP;
- K_C = dnGj1XXqdebDvj8LK3Ch4YtGsUw95PGcHr6XRbqfKhZ.

Grazie alle chiavi generate (é riportata solo quella pubblica poiché quella privata non risulta mai visibile all'interno delle transazioni) é stata effettuata una transazione CREATE da parte di P_A tramite il driver Javascript. A seguito della transazione, contattando l'API tramite il programma *Postman*, si riceve un array contenente tutte le transazioni che sono state effettuate tramite K_A (Figura 2.15); nel nostro caso l'array contiene una sola transazione.

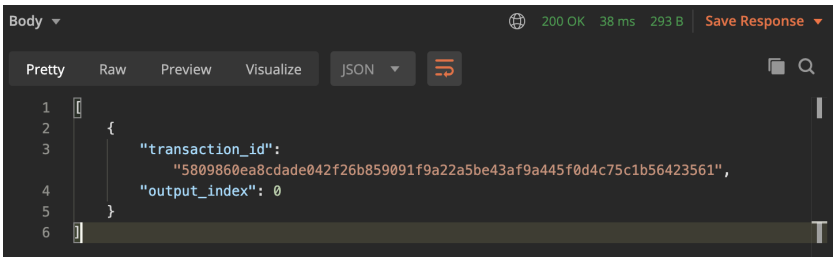


Figura 2.15: Risultato della richiesta di tutte le transazioni associate ad una chiave pubblica

Da questo array é possibile recuperare l'ID della transazione appena effettuata e ricercarla tramite la route apposita. Il risultato della ricerca é l'oggetto JSON seguente (alcuni campi non importanti sono stati sostituiti con"..." per una maggiore compattezza):

```
{
"inputs": [
    {
    "owners_before": [
        "3LrM7Pv9RrNvqD12RUgYJwQR86hG8apRne9ADGbUy4Ut"
    ],
    "fulfills": null,
    "fulfillment": "..."
    }
],
```

```
"outputs": [
    {
    "public_keys": [
        "3LrM7Pv9RrNvqD12RUgYJwQR86hG8apRne9ADGbUy4Ut"
    ],
    "condition": {
        ...
    },
    "amount": "1"
    }
],
"operation": "CREATE",
"metadata": {
    "Operation": "CAR CREATION"
},
"asset": {
    "data": {
        "date": "1609254907779",
        "asset-name": "red-car",
    }
},
"version": "2.0",
"id": "5809860ea8cdade042f26b859091f9a22a5be43af9a445f0d4c75c1b56423561"
}
```


In questo esempio la transazione rappresenta come asset un'automobile, ed é stato aggiunto anche un campo `"date"` per imprimere la data in cui é avvenuta la creazione dell'asset. Come si puó vedere dall'oggetto JSON l'array di output contiene la chiave K_A e, essendo una transazione di CREATE, anche quello di input contiene la stessa chiave, ad indicare che P_A é sia il proprietario attuale che il precedente; l'ID della transazione viene visualizzato come ultimo campo.

Successivamente é stata eseguita una operazione TRANSFER, anche questa mediante il driver per Javascript, per trasferire l'asset da P_A a P_B; é importante notare come per effettuare questo passaggio sia sufficiente conoscere la chiave pubblica di K_B e che quindi l'utente B non possa in alcun modo pratico evitare di ricevere una TRANSFER. Dopo il trasferimento, che avviene solo se P_A fornisce la propria chiave privata, ricercando le transazioni legate a K_B é possibile risalire all'ID della transazione di TRANSFER e visualizzarla sempre tramite *Postman*. Il risultato é il seguente:

```
{
"inputs": [
    {
    "owners_before": [
        "3LrM7Pv9RrNvqD12RUgYJwQR86hG8apRne9ADGbUy4Ut"
    ],
    "fulfills": {
        ...
    },
    "fulfillment": "..."
    }
],
"outputs": [
    {
    "public_keys": [
        "GdAxyeikX7B4tf41mQww7dr65KFmeNv54kP3syGvFUwP"
    ],
    "condition": {
        ...
    },
    "amount": "1"
    }
],
"operation": "TRANSFER",
"metadata": {
    "Operation": "Car transfer from A to B"
},
"asset": {
    "id": "5809860ea8cdade042f26b859091f9a22a5be43af9a445f0d4c75c1b56423561"
},
"version": "2.0",
"id": "d049478a3dc42c9df1baf0bee0be28ed92857b65811ea54babc4bff5b7e96655"
}
```


A seguito del trasferimento inputs contiene P_A, cioé la chiave pubblica del precedente proprietario, e outputs contiene P_B, chiave pubblica del proprietario attuale dell'asset. All'interno del campo asset viene invece riportato semplicemente l'ID della transazione CREATE originale.
Si puó notare come, mentre il campo asset non sia editabile durante una TRANSFER, lo sia invece il campo metadata, che puó essere usato per alle-

gare informazioni al trasferimento di proprietá.

Come ultimo test é stata effettuata un'altra operazione di TRASFER, firmando la nuova transazione mediante k_B e trasferendo l'asset a P_C tramite la sua chiave pubblica. Il risultato é strutturalmente analogo a quello di cui sopra, dove in inputs troviamo P_B e in outputs P_C. Notevole é il fatto che all'interno del campo asset non avvenga alcuna modifica e che quindi il riferimento sia sempre alla transazione CREATE originale. Per completezza viene riportato l'oggetto JSON di seguito.


```
{
"inputs": [
    {
        "owners_before": [
            "GdAxyeikX7B4tf41mQww7dr65KFmeNv54kP3syGvFUwP"
        ],
        "fulfills": {
            ...
        },
        "fulfillment": "..."
    }
],
"outputs": [
    {
    "public_keys": [
        "dnGj1XXqdebDvj8LK3Ch4YtGsUw95PGcHr6XRbqfKhZ"
    ],
    "condition": {
        ...
    },
    "amount": "1"
    }
],
"operation": "TRANSFER",
"metadata": {
    "Operation": "Car transfer from B to C"
},
"asset": {
    "id": "5809860ea8cdade042f26b859091f9a22a5be43af9a445f0d4c75c1b56423561"
},
"version": "2.0",
```

"id": "60c8367d679411dea19be7f37070fb3eaaf1088713f193f1e8545f9aa7e4b37a"
}

2.3.4 Caratteristiche database-like esposte da bigchainDB

Come anticipato, oltre alle caratteristiche base della *blockchain*, *bigchainDB* espone alcune funzionalitá base dei DB tradizionali.

Primo tra tutti il sistema é realizzato per garantire un alto valore di *TPS*, e quindi per processare migliaia di transazioni al secondo. Nonostante il progetto sia in continuo sviluppo la documentazione che si trova, compreso il whitepaper, non riporta test di performance effettuati direttamente dai creatori di *bigchainDB*, che mostrano pertanto i risultati di test effettuati su altri progetti utilizzanti *Tendermint*; poiché il throughput del sistema é direttamente legato alla velocitá di raggiungimento del consenso e a quella della trasmissione del risultato ai nodi, il collo di bottiglia si trova al livello di questo software. Considerando alcuni test effettuati sul progetto *Cosmos* [14], atto a realizzare un network *BFT* di *blockchain* indipendenti tramite appunto *Tendermint*, il sistema di consenso si é mostrato in grado di gestire piú di mille transazioni al secondo, lavorando su 64 nodi distribuiti in 7 datacenter in 5 continenti, quindi con architettura estremamente distribuita.
Anche dal punto di vista della latenza il sistema si mostra estremamente efficiente, con tempi di attesa di qualche secondo, o meno, per inserire le transazioni all'interno del blocco e registrarlo sulla *blockchain*; é da sottolineare come inoltre *Tendermint* non consideri la possibilitá di effettuare delle fork, quindi il tempo necessario per considerare un blocco valido dopo che é stato proposto al sistema é esattamente pari al tempo necessario a registrarlo sulla *blockchain* in modo permanente, in quanto non potrá piú essere considerato scorretto e quindi sostituito in futuro.

Altra caratteristica interessante che *bigchainDB* eredita dai DB é la possibilitá di effettuare delle query al suo interno per recuperare dati conservati; é possibile ricercare all'interno del DB transazioni, asset, metadata o blocchi interi, tutti conservati come stringhe *JSON*. I nodi del network sono in grado, tramite la propria API, di decidere in maniera autonoma quanto permettere all'utente esterno di addentrarsi all'interno del DB e come implementare queste queries nello specifico.

2.3.5 Considerazioni finali su bigchainDB

BigchainDB implementa una vera e propria *blockchain* con una struttura molto piú semplice di quella di *Fabric* e molto meno costosa dal punto di vista computazionale di *Bitcoin*. La sua natura la rende adatta alla realizzazione di *blockchain* sia pubbliche che private, e garantisce un alto livello di personalizzazione della rete, definendo politiche di accesso anche complesse tramite l'API server interposto tra il nodo e l'utente finale.

Il motivo principale che spinge all'uso di *bigchainDB* é la semplicitá di uso della catena di proprietá delle transazioni, che permette di abilitare una serie di modelli di lavoro che fanno tutti capo al cosiddetto "digital-twin". Con questo termine si intende la rappresentazione digitale di un oggetto fisico che puó essere tracciata verificandone l'autenticitá e la provenienza in ogni momento e per tutto il suo "ciclo di vita".
Grazie alla facilitá della costruzione degli asset digitali, *bigchainDB* si dimostrerá piú avanti in questo progetto la scelta definitiva per la gestione della proprietá di oggetti e documenti, oltre che per la realizzazione di un vero e proprio *ledger* distribuito per la gestione della proprietá intellettuale.

Nonostante queste sue caratteristiche si é scelto di realizzare la prima implementazione del progetto di cui si discuterá in seguito mediante *Fabric*, poiché, avendo questo necessitá di essere presentato di fronte a gruppi di aziende, si é supposto piú utile un sistema in grado di fornire il tipo di flessibilitá e controllo ibrido che i canali di *Fabric* possono garantire.

Inoltre, per l'uso come vera e propria *blockchain* pubblica, *bigchainDB* continua a risentire della difficoltá pratica di decentralizzare il controllo in maniera tale da ritenerlo sufficientemente distribuito. Poiché per i fini industriali non sono spesso accettabili i costi necessari a sostenere questo tipo di decentralizzazione si é preferito, in un primo momento, affiancare a *Fabric* una struttura giá preesistente e decentralizzata, quale *Iota*, che verrá illustrata nella prossima sezione.

2.4 IOTA: la DLT pensata per l'IoT che possiede le caratteristiche della blockchain

2.4.1 DLT e blockchain, similitudini e differenze

Il termine *DLT* indica in senso generale un registro distribuito che puó essere usato per conservare informazioni e scambiare valore in modo decentralizzato. É importante sottolineare la differenza tra database distribuito e *DLT*: il primo rappresenta un sistema di conservazione dell'informazione decentralizzato su piú nodi, ognuno dei quali mantiene l'intera copia del contenuto, per garantire "fault tolerance" and "disaster recovery", cioé la capacitá del sistema di resistere ad eventi, anche gravi, che ne corrompano una parte; il secondo rappresenta invece vero e proprio *ledger* ove si conservano informazioni, basato su un algoritmo di consenso. In una *DLT* non é possibile inserire direttamente ogni tipo di informazione ma é necessario che tutti i nodi concordino sulla validitá della stessa, partecipando al processo di verifica.
Questa caratteristica accomuna le *DLT* alla *blockchain* e, in termini semplificati, é possibile definire la seconda come una implementazione della prima, realizzata mediante una specifica struttura dati, la "catena di blocchi" appunto. Altro punto in comune é l'ampio uso della crittografia, indispensabile per implementare gli algoritmi di consenso.

Le caratteristiche fondamentali che sono presenti in tutte le *DLT* sono quindi:

- presenza di una struttura decentralizzata sulla rete su cui vengono conservate informazioni;
- uso di un algoritmo di consenso;
- aggiornamento del database in maniera indipendente da parte di ogni nodo, senza autoritá centrale.

Le strutture propriamente definite *blockchain* aggiungono poi a queste caratteristiche la possibilitá di creare "owner-controlled assets" e di trasferirli tra utenti in maniera diretta.

Anche per le *DLT* abbiamo la distinzione tra permissioned e permissionless. Nel primo caso, quando un utente vuole entrare a far parte del sistema deve identificarsi ed essere accettato esplicitamente; per registrare poi una

transazione l'utente la sottopone alla rete, la quale decide (tramite un algoritmo solitamente piú semplice del caso permissionless) se considerarla valida e aggiungerla al registro. Nel secondo caso, invece, l'algoritmo di consenso é piú complesso (per esempio un *PoW*) e, sebbene ogni utente possa richiedere direttamente di effettuare una transazione, l'algoritmo si preoccupa di intercettare comportamenti malevoli atti a influenzare negativamente il funzionamento del sistema.

2.4.2 Il concetto alla base di Iota: per chi é stata creata

IOTA si presenta come uno strumento di scambio di valore specifico per micro-transazioni e per l'utilizzo da parte di dispositivi *IoT*. L'implementazione comprende una criptovaluta, con lo stesso nome del progetto, che puó essere scambiata sulla rete in maniera completamente "feeless" e con alti valori di *TPS*.

L'idea alla base del progetto *IOTA* [15] é quella di realizzare un "trust layer" (letteralmente layer di fiducia) orizzontale sulla rete internet, utilizzabile da qualsiasi tipo di dispositivo che sia in grado di connettersi alla rete internet. Tramite questo layer, le cui specifiche saranno spiegate in seguito, é possibile usare la rete come fonte di fiducia per i dati oppure per scambiare valore tramite la criptovaluta.
Come evidenziato dalla documentazione il progetto é pensato per l'uso da parte di:

- Utenti che abbiano perso fiducia nei sistemi centralizzati e vogliano affidarsi ad architetture basate sul consenso;
- Utenti che necessitino di effettuare transazioni istantanee, sicure e senza commissioni;
- Utenti che ricercano la possibilitá di rendere sicuri e trasparenti i dati, potendo garantirne l'immutabilitá nel tempo.

Usando l'ecosistema di IOTA viene eliminato dall'equazione il concetto delle commissioni applicate alle transazioni. Infatti, *Bitcoin*, garantisce ai miners una commissione ogni volta che riescono ad inserire un blocco sulla catena, in aggiunta alla creazione della nuova moneta; nel momento in cui non saranno piú minati nuovi *Bitcoin* queste commissioni rappresenteranno l'incentivo per i miner ad agire in maniera virtuosa.
Con *IOTA* si affronta il problema dei micro-pagamenti, sempre piú comuni oggi ma ancora non incentivati dalle *blockchain* tradizionali, in quanto spesso la commissione sulla transazione risulta superiore all'effettiva somma da

scambiare; grazie alla rete *IOTA* le commissioni sono del tutto inesistenti, e non sono necessari nemmeno incentivi ai miners poiché, nel senso proprio del termine, questi non esistono. Mentre infatti in *Bitcoin* sono presenti due tipologie di partecipanti, distinti tra chi approva transazioni e crea valuta e chi invece effettua le transazioni, in *IOTA* queste due operazioni sono effettuate dallo stesso utente. L'utente, per poter effettuare una propria transazione, deve confermarne un determinato numero, partecipando quindi attivamente al processo di consenso; di conseguenza la transazione appena effettuata sará inserita nel *ledger* non appena un altro utente, per effettuare la propria transazione, la confermerá, e cosí via.

2.4.3 La rivoluzione di IOTA: il Tangle

L'architettura della rete *IOTA* (Figura 2.16) é composta da tre componenti fondamentali:

1. Nodi: i componenti veri e propri della rete a cui vengono indirizzate le transazioni;

2. Clients: gli utenti della rete che inviano le transazioni ai nodi per farle registrare sulla rete;

3. Tangle: il vero e proprio *ledger* distribuito che viene replicato su tutti i nodi IOTA.

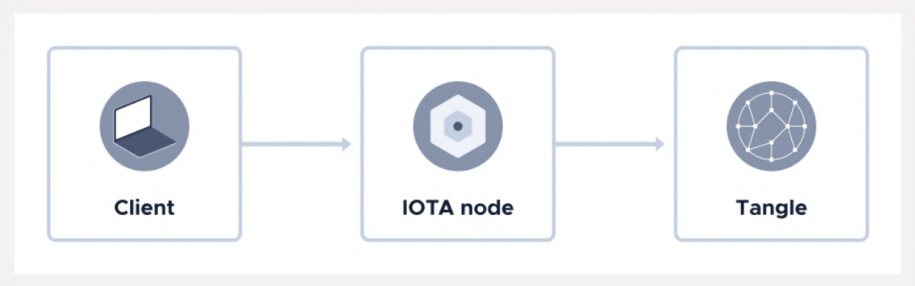


Figura 2.16: I componenti dell'ecosistema IOTA [16]

Secondo il whitepaper di *IOTA* [17] il *Tangle* rappresenta l'evoluzione naturale della *blockchain* e la sua versione evoluta per il salvataggio delle transazioni.

Il concetto alla base del *Tangle* non é stato creato da *IOTA* ma, con questo termine, viene indicato un "grafo aciclico diretto" (DAG). Questa particolare struttura di grafo presenta due caratteristiche fondamentali per comprendere il funzionamento di *IOTA*:

- Grafo aciclico: rappresenta un grafo dove é impossibile, scelto un vertice dello stesso, tornarvici percorrendo degli archi. Non vi sono quindi percorsi chiusi tra vertici;
- Grafo diretto: ogni arco ha una e una sola direzione e non puó essere percorso in senso opposto.

All'interno del *Tangle* le transazioni costituiscono il set di vertici del grafo, che é il *ledger* su cui sono salvate e che viene condiviso da tutti i nodi. Da qui in poi si definiranno "nodi" i computer o server deputati al mantenimento dell'infrastruttura, mentre "vertici" le unitá base del grafo collegate dagli archi, e conseguenza anche le transazioni stesse.

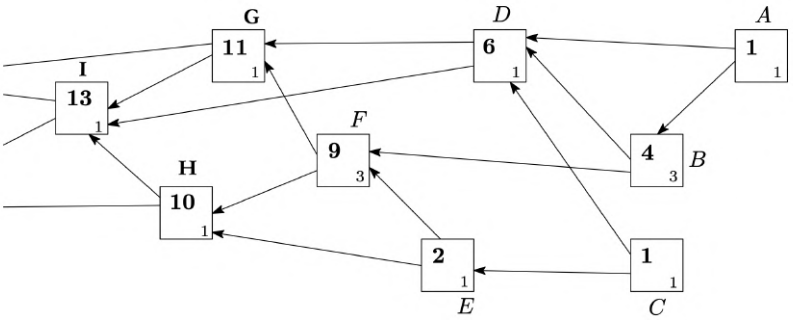


Figura 2.17: Rappresentazione schematica di un Tangle [17]

Per costruire il grafo si agisce nel seguente modo: quando una transazione A viene inviata nel sistema essa deve approvare due transazioni precedenti; l'approvazione avviene mediante la creazione di un arco che collega A ad altre due transazioni. Quando tra due vertici A e B é presente un arco si dice che la transazione da cui questo parte conferma "direttamente" la transazione in cui l'arco arriva; é possibile anche che la conferma sia "indiretta", qualora sia presente un cammino di lunghezza almeno due che collega A e B. Nel caso riportato in Figura 2.17 A conferma direttamente B e D, e indirettamente conferma anche F,G,I,H.
L'unica eccezione a questa struttura é data dalla transazione di genesi, che

viene inserita come prima transazione del *Tangle* alla creazione dello stesso, e che quindi é direttamente o indirettamente approvata da tutte le altre transazioni.

Quando un nodo effettua una transazione sceglie quindi due transazioni da approvare, seguendo un algoritmo che verrá illustrato successivamente, controlla se queste sono valide e risolve un *PoW*, esattamente come nel caso di *Bitcoin*. Sul *Tangle* sono poi riporati dei valori numerici, che rappresentano indici utilizzati per la definizione della prioritá delle transazioni, e cioé quanto il nodo che sta cercando di validare la transazione é disposto a spenderci in termini di potenza computazionale. Questo valore, posto in basso a destra, é definito *peso* della transazione ed é sempre un valore intero positivo; viene utilizzato dalla rete per comprendere quanto la transazione sia importante rispetto ad un'altra.
Accanto al *peso* della transazione viene definito anche il suo *peso cumulativo*, cioé la somma del proprio *peso* e di quello di tutte le transazioni che direttamente o indirettamente la confermano. Questo valore é riportato in alto a sinistra in Figura 2.17 e viene aggiornato ogni volta che viene aggiunta una transazione. Come é possibile vedere la transazione B, il cui *peso* é pari a 3, ha *peso cumulativo* pari a 4 in quanto viene confermata solo da A. Nel momento in cui la situazione diventa quella di Figura 2.18, aggiugnendo una transazione X che conferma sia A che C, il peso cumulativo di B diventa pari a 7, poiché il peso di X é pari a 3.

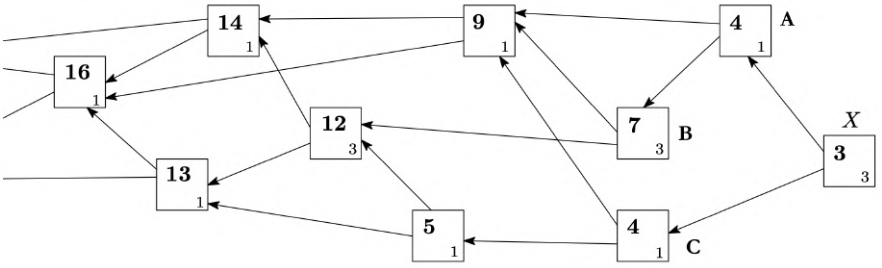


Figura 2.18: Evoluzione del Tangle con l'inserimento di X [17]

2.4.4 Algoritmi di selezione delle transazioni da approvare

Nella terminologia di *IOTA* vengono indicate come *tips* le foglie del grafo, cioé le transazioni che ancora non hanno ricevuto alcuna approvazione e quindi verso cui non é indirizzato alcun arco. In Figura 2.18 la transazione X é

una *tip* e approva le transazioni A e C. La chiave dietro la struttura di *IOTA* é proprio l'algoritmo che viene utilizzato per scegliere le *tip* da approvare da parte delle nuove transazioni.

Un concetto importante per comprendere l'algoritmo di scelta delle *tips* é quello del processo di Poisson, che modella una situazione nella quale abbiamo eventi del tutto indipendenti che avvengono in maniera continua nel tempo. Questo tipo di processo segue la distribuzione di Poisson, che definisce una serie di eventi che si verificano successivamente all'interno di un determinato intervallo di tempo, sapendo che mediamente il numero é pari a λ. La probabiliá di avere n eventi in un determinato intervallo di tempo, quando mediamente ne abbiamo λ viene data come:

$$P_\lambda(n) = \frac{\lambda^n}{n!} \cdot e^{-\lambda} \tag{2.6}$$

Questo processo viene usato per rappresentare una serie consequenziale di eventi random che avvengono all'interno di un intervallo di tempo prefissato, ed é utilizzabile per rappresentare l'arrivo delle transazioni al *Tangle*, supponendo quindi che vi possano essere intervalli in cui ne arrivano un numero maggiore e intervalli in cui non ne arrivano affatto.

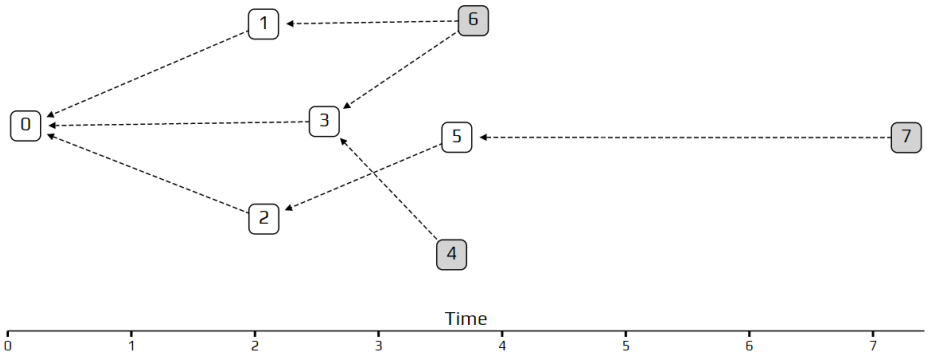


Figura 2.19: Arrivo delle transazioni secondo un processo di Poisson [18]

Modellando il sistema in questo modo possiamo supporre che, sia l'intervallo temporale pari a 1 secondo e il valore di $\lambda = 2$ allora mediamente occorranno 50 secondi per effettuare 100 transazioni.

Il *Tangle* di *IOTA* inoltre é costruito per inserire un delay pari ad h ad ogni transazione; questo delay indica quanto tempo la transazione necessita per essere propagata all'intera rete dopo essere stata effettuata, e quindi

quanti slot temporali é necessario attendere prima di poterla confermare. Senza la presenza di questo delay si presenterebbe la possibilitá per cui, a bassi valori di λ, il DAG diventi una semplice catena dove ogni transazione vede come unica *tip* la transazione subito precedente e conferma solo quella poiché non ne trova altre effettuate nello stesso istante temporale che non siano confermate.

Possiamo quindi supporre che per decidere a quale *tip* collegare una nuova transazione (aggiungendo quindi un arco dalla nuova alla *tip*) si possa seguire un cammino che parta dalla transazione di genesi e percorra il grafo per valori crescenti dell'asse dei tempi; questo viene detto "unweighted random walk". Posizionando un "walker" alla transazione di genesi percorriamo quindi il cammino all'indietro scegliendo ogni volta l'arco da seguire; nei casi in cui ad un vertice arrivi piú di un arco, e quindi sia confermato da piú di una transazione, si suppone inizialmente che siano equiprobabili.
Nel caso riportato in Figura 2.20, partendo dalla tranzazione 0 ci si sposta prima su 1 e poi su 2 con probabilitá pari ad uno; da 2 abbiamo poi tre transazioni in cui spostarci per cui la probabilitá di ogni arco é pari ad 1/3.

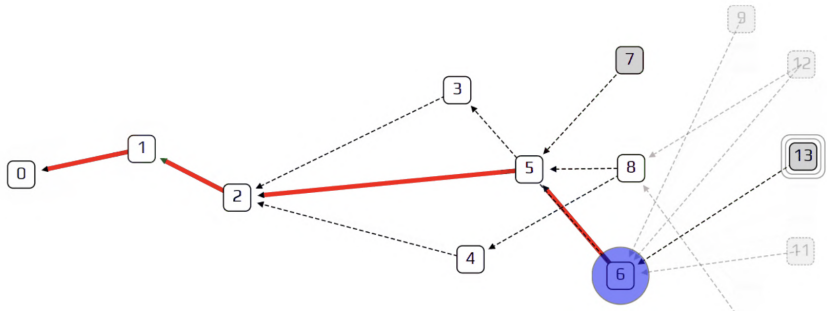


Figura 2.20: Percorso dalla genesi ad una tip seguendo un unweighted random walk [18]

Supponendo che venga scelta 5, l'unica transazione verso cui ci si puó spostare sará quindi 6, che rappresenta anche una *tip*, a cui collegare la generica transazione 13 in arrivo, in quanto le transazioni 9,12,11 sono ancora invisibili a 13 a causa del delay introdotto.

Alternativa a questo algoritmo potrebbe anche essere una selezione completamente casuale della *tip* da approvare, con le nuove transazioni che ne approvino una completamente causale. Questo porta tuttavia al problema delle cosiddette "lazy-tips", cioé quelle che approvano transazioni molto vecchie invece di quelle recenti (in Figura 2.21, 14 é una lazy-tip); questo, usando

un algoritmo casuale, risulta equiprobabile rispetto all'approvazione di transazioni recenti. Una *lazy-tip* non partecipa attivamente al mantenimento del network in quanto non approva nuove transazioni ma semplicemente immette dati propri sfruttando informazioni molto vecchie.

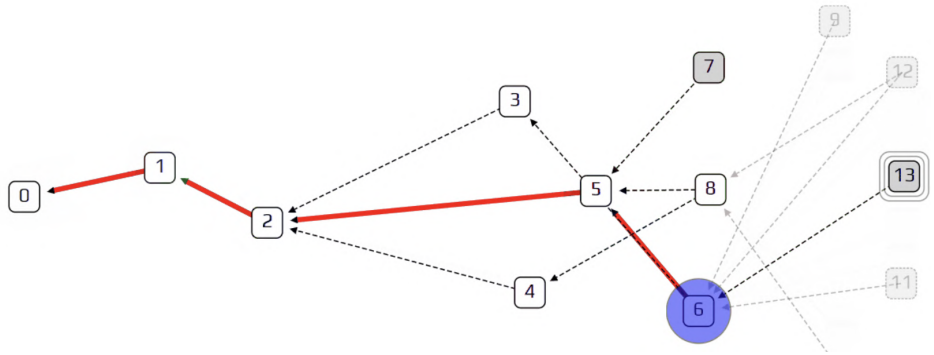


Figura 2.21: Creazione di una lazy-tip a causa della scelta di un algoritmo di selezione random [18]

Nel caso di un algoritmo casuale una *lazy-tip* ha la stessa probabilitá di essere confermata di ogni altra, compromettendo quindi il corretto funzionamento del sistema. In aggiunta, nel caso in cui venga usato un algoritmo di *unweighted random walk*, una *lazy-tip* ha addirittura una probabilitá piú alta di essere approvata rispetto ad una *tip* recente.

Una soluzione al problema delle *lazy-tips* potrebbe consistere nell'imporre la conferma solo di transazioni recenti, ma questo sarebbe in contrasto con il concetto di decentralizzazione. La soluzione consiste nel costruire il sistema in modo che il comportamento sia scoraggiato in maniera automatica; per effettuare questa operazione viene utilizzato il *cumulative weight* delle transazioni.
Il *random walk* viene quindi polarizzato introducendo una regola secondo la quale é piú probabile includere nel cammino transazioni con un alto *peso cumulativo* rispetto alle altre. Per un questione di semplicitá supponiamo che ogni transazione abbia *peso* pari ad 1, in modo che il suo *peso cumulativo* sia pari al numero di transazioni che la approvano piú uno. Nell'esempio in Figura 2.22 la transazione 16 é una *lazy-tip* per cui, quando il *random walker* giunge a 7 deve decidere tra 16 e 9, la situazione sará la seguente: poiché 16 ha peso cumulativo pari ad uno mentre 9 pari a sette sará molto piú probabile che il cammino proceda in quella direzione.

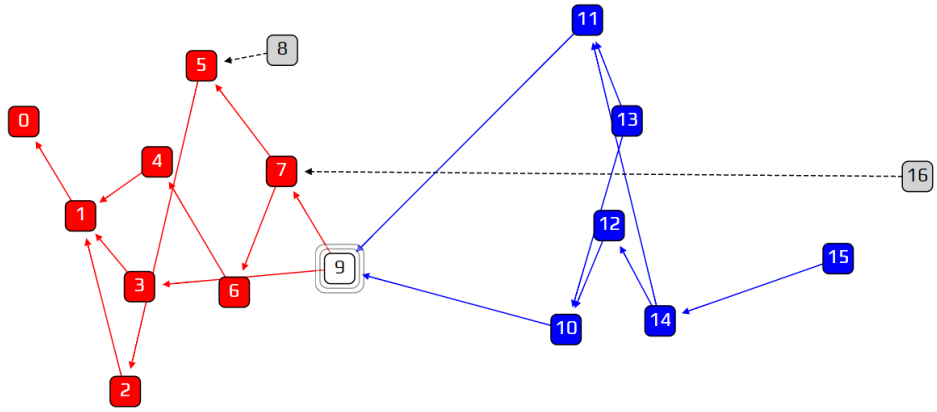


Figura 2.22: Percorso dalla genesi ad una tip seguendo un weighted random walk [18]

Per decidere quanto sia maggiore questa proababilitá viene usato un parametro α, che indica sostanzialmente quanto é importante scegliere un vertice con peso cumulativo rispetto ad uno inferiore. Se questo valore risulta troppo alto si ha il caso in cui viene sempre scelta la transazione con peso maggiore, inserendo nel grafico una serie di transazioni a peso inferiore che non verranno mai approvate; al contrario se si sceglie un parametro troppo basso si ritorna al caso dell'*unweighted random walk* e il numero delle *lazy-tips* diventa paragonabile, su base statistica, a quelle normali.
Un metodo di questo tipo, che si basa su una regola che stabilisca la probabilitá di ogni passo in un *random walk*, é definito "Markov Chain Monte Carlo Technique" [19] e non verrá ulteriormente approfondito.

2.4.5 Problema del double-spending e raggiungimento del consenso

Poiché *IOTA* é utilizzato per scambiare valore é di grande interesse la risoluzione del problema del "double-spending". Prima di tutto é necessario puntualizzare come *IOTA* non abbia il concetto di "mining", per cui tutti gli *IOTA* esistenti sono stati creati con la prima transazione di genesi del *Tangle*, poi distribuiti ai fondatori sui propri account in base alla quantitá di fondi inseriti nel progetto.

Prendiamo un utente generico, $ALICE$, il quale vuole trasferire $10i$ ($IOTA$) a BOB, il cui bilancio attuale é di $0i$; $ALICE$ dovrá semplicemente effettuare una transazione e inserirla nel *Tangle*, eseguendo l'algoritmo di selezione

della *tip*, per approvare due transazioni e inserire quindi la sua sul *DAG*. Supponiamo ora che un terzo utente, $CHARLIE$, voglia fare una sua transazione e che per farla l'algoritmo selezioni la transazione fatta in precedenza da $ALICE$ come transazione da approvare; il compito di $CHARLIE$ é quello di confermare che $ALICE$ abbia effettivamente i $10i$ che ha inviato a B. Questo compito deve essere eseguito correttamente, altrimenti, qualora $CHARLIE$ approvasse una transazione fraudolenta, si ritroverebbe nella condizione in cui nessun utente approverebbe poi la sua.
Per confermare la transazione di $ALICE$, il generico utente $CHARLIE$ deve ricostruire, lungo il *Tangle*, il percorso della valuta che ha portato $ALICE$ ad avere i $10i$; per farlo inserisce in una lista tutte le transazioni approvate direttamente o indirettamente dalla transazione di $ALICE$, fino alla genesi. Una volta creata questa lista, se $ALICE$ risulta avere l'ammontare di token che ha richiesto di scambiare la sua transazione viene approvata da $CHARLIE$.

Il caso piú semplice di tentativo di commettere un'operazione fraudolenta consiste nel cercare di spendere piú di quanto si possiede; se per esempio $ALICE$ cercasse di scambiare con BOB $100i$, avendone soltanto dieci la sua transazione non verrebbe mai approvata da un altro utente in quanto, nell'effettuare il controllo sul percorso della valuta, si renderebbe conto che $ALICE$ non puó possedere quell'ammontare di i. L'utente che conferma le transazioni é infatti disincentivato nell'approvare una transazione fraudolenta, poiché la sua non verrebbe poi approvata a sua volta, creandogli uno svantaggio diretto.

Nel protocollo di *IOTA* tuttavia, per velocizzare le transazioni, ognuna ne conferma due precedenti e, nel farlo, controlla ciascuna di esse per verificarne la validitá. Questo meccanismo apre, a livello teorico, la possibilitá di effettuare un double-spending: infatti, qualora $ALICE$ inserisse due transazioni identiche in cui scambia $10i$ con BOB, avendone soltanto 15 nel proprio bilancio, ognuna delle due risulterebbe corretta al momento dell'inserimento, in quanto $ALICE$ possiede abbastanza valuta per effettuarne ognuna singolarmente, anche se non entrambe. Durante la fase di controllo peró $CHARLIE$ é in grado di rendersi conto che $ALICE$ ha effettuato il "double-spending" in quanto, dopo aver approvato la prima, all'approvazione della seconda il bilancio totale di $ALICE$ diventa negativo.
In questa situazione si creano quindi due rami del *Tangle*, che non sono compatibli tra loro, e uno dei due deve essere abbandonato, lasciando la transazione non approvata. L'utilizzo del *weighted walk* risolve questo problema poiché sicuramente uno dei due rami crescerá piú in fretta e l'altro,

che diventerá piú "leggero", verrá abbandonato.

Il compromesso per il funzionamento di questo meccanismo é che una transazione non sia confermata immediatamente dopo essere stata effettuata, anche se giá approvata da qualche utente, ma solo dopo il raggiungimento della cosiddetta "confirmation confidence". Questo termine indica il livello di accettazione della transazione da parte del resto del *Tangle* e viene calcolato con il seguente procedimento:

1. L'algoritmo di selezione della *tip* viene eseguito 100 volte;
2. Si conta il numero di volte in cui l'algoritmo approva la transazione scelta, indicandolo con β;
3. La *confirmation confidence* della transazione generica é quindi pari a $\beta\%$.

Un modo per valutare la *confirmation confidence* di una transazione é quindi contare la percentuale di *tips* che la approvano, direttamente o indirettamente, dando piú importanza alle *tips* piú probabili. Quando una transazione raggiunge un valore di *confirmation confidence* abbastanza alto, impostato solitamente al 95%, viene considerato altamente improbabile che la transazione sia eliminata dall'algoritmo del consenso; é importante soffermarsi sul fatto che, esattamente come nel caso di *Bitcoin*, tutte le considerazioni vadano sempre fatte nell'ordine dell'"improbabile", ma non impossibile, in quanto $ALICE$ potrebbe in ogni caso avere abbastanza potenza computazionale da riuscire ad ingannare il sistema (esattamente come nel caso dell'attacco 51% possibile per *Bitcoin*).

Potenzialmente infatti $ALICE$ potrebbe effettuare una transazione T_{AB} a BOB per acquistare un T-Rex (l'esempio riportato proviene dalla documentazione ufficiale di *IOTA* [20]) e attendere che questa venga confermata, raggiungendo la *confirmation confidence*, per poi richiedere a BOB di fornirle effettivamente quanto acquistato. Contemporaneamente $ALICE$ effettua anche una transazione T_{AC} a $CHARLIE$, acquistando un altro T-Rex, con la stessa moneta spesa prima con BOB: in questo caso ci troviamo di fronte ad un double-spending e il sistema dovrá decidere quale ramo del *Tangle* abbandonare. Teoricamente, avendo T_{AB} raggiunto la *confirmation confidence*, la transazione con $CHARLIE$ dovrebbe essere considerata non valida e quindi non approvata da nuove *tips*, portando all'abbandono di quel ramo e al non raggiungimento della *confirmation confidence*. Tuttavia, se $ALICE$ possedesse abbastanza capacitá di calcolo, sarebbe in grado di usare la T_{AC} per

confermare due vecchie transazioni e poi continuare ad aggiungere centinaia di nuove transazioni sperando che approvino T_{AC} aumentando il *cumulative weight* del nuovo ramo fraudolento. Il risultato di questa operazione potrebbe portare al raggiungimento della *confirmation confidence* per T_{AC}, calando contemporaneamente quello di T_{AB} fino all'abbandono del ramo e quindi al raggiungimento del valore dello 0%. In Figura 2.23 é riportato questo caso, supponendo l'asse dei tempi muoversi verso il basso; una volta raggiunta la *confirmation confidence* su T_{AC} *BOB* si troverebbe con una transazione a lui diretta che non viene piú approvata e quindi senza la possibilitá di usare gli *IOTA* ricevuti, nonstante abbia giá provveduto a consegnare l'articolo richiesto.

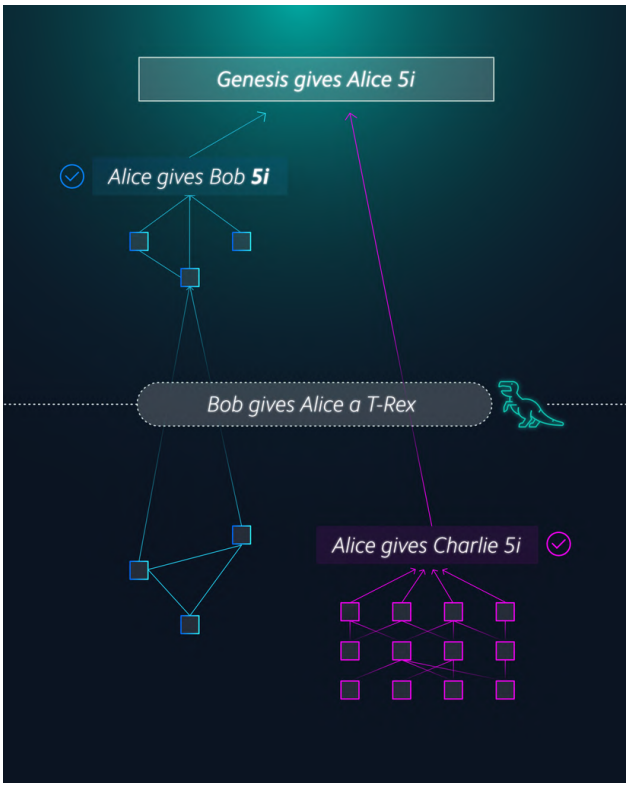


Figura 2.23: Evoluzione del Tangle a seguito del double-spending [20]

Poiché la condizione alla base della riuscita di questa frode é che *ALICE* da sola possieda piú potenza computazionale di tutto il resto della rete, il problema non si pone in un sistema maturo e attivo, ma é tutt'altro che ignorabile per la struttura attuale di *IOTA*, che presenta ancora troppe poche transazioni nel sistema per rendere l'attacco impossibile.

Per questo motivo nell'ecosistema di *IOTA* attuale viene implementato temporaneamente un meccanismo di consenso differente che assicura l'impossibilitá di effettaure *double-spending*. Ogni due minuti la rete *IOTA* genera una transazione, detta *milestone*, tramite un componente chiamato "Coordinator", e tutte le transazioni che vengono approvate da questa assumono automaticamente *confirmation confidence* pari al 100%. Usando questo sistema la seconda transazione di $ALICE$ non avrebbe mai raggiunto l'approvazione.

L'aggiunta del coordinatore riduce parzialmente il livello di decentralizzazione della rete *IOTA* ed é necessario quindi che, ad un certo punto, questo sistema venga deprecato e la rete venga lasciata funzionare in maniera del tutto autonoma. Questo momento é fissato all'incirca nella prima metá del 2021 [21], con il cosiddetto "Coordicide" [22], che avverrá quando la rete sará considerata abbastanza matura da garantire il funzionamento dell'algoritmo di consenso distribuito standard. Dopo questa operazione la rete diventerá al 100% decentralizzata e sará lasciata al proprio sviluppo in maniera del tutto indipendente.

2.4.6 Fruibilitá di IOTA per le applicazioni dell'Internet of Things: CCurl PoW

La fruibilitá dell'ecosistema di *IOTA* deriva direttamente dal tipo di algoritmo PoW che viene utilizzato per inserire le transazioni all'interno del sistema. In *Bitcoin* il PoW viene usato per scoraggiare gli utenti a inserire transazioni fraudolente, costringendoli a consumare numerose risorse per generare un blocco; oltre a questo é anche fondamentale per limitare il numero di blocchi inseriti nella catena, mantenendone il rateo costante, visto che ognuno di questi genera nuovi *Bitcoin*. In *IOTA* l'uso eccessivo di risorse é incompatibile con la logica del sistema e, inoltre, non é presente alcun tipo di generazione di valuta poiché ogni transazione costituisce un nuovo vertice del *Tangle* e vengono scambiati solo gli $IOTA$ generati nella "genesi". Il PoW viene quindi utilizzato per impedire lo spamming eccessivo di transazioni da parte degli utenti, motivo per cui é stato scelto un algoritmo direttamente implementabile su dispositivi di piccole dimensioni e senza elevate capacitá computazionali, e che sopratutto non richieda un eccessivo consumo di risorse.

Il tipo di PoW scelto é il "Curl-PoW" [24]: per implementarlo ogni transazione riferita allo stesso scambio di valore viene inserita dal dispositivo all'interno di un "bundle", che viene poi firmato con la propria chiave priva-

ta. Il PoW viene poi eseguito per ogni transazione presente nel bundle.

Per comprendere al meglio questo tipo di funzione é necessario introdurre una logica ternaria, che viene modificata da *IOTA* [25], sostituendo ai classici 0,1,2 i valori 0,1 e -1; al posto dei *bit* abbiamo quindi i *tri* e al posto dei *bytes* i *trytes*, composti da tre *trit*. Per la progettazione di *IOTA* si é scelto di rappresentare il *tryte* mediante le 26 lettere dell'alfabeto, aggiungendo il numero nove (per un totale di $3^3 = 27$ simboli), quindi il set completo é:

$$9ABCDEFGHIJKLMNOPQRSTUVWXYZ \tag{2.7}$$

La conversione tra *bytes* e *trytes* avviene considerando che un *tryte* puó rappresentare i numeri da -13 a 13, essendo composto solo da tre *trit*, al contrario degli otto *bit* che compongono un *byte*; in un *byte* sono presenti quindi circa tre *trits*, in grado di rappresentare 3^5 valori:

$$n = \frac{8}{\log_2 3} \cong 5 \tag{2.8}$$

Mentre con un *byte* rappresentiamo $2^8 = 256$ valori, con cinque *trits* ne rappresentiamo $3^5 = 243$.

Il *Curl-PoW* viene fatto attraverso una funzione di hashing chiamata Curl (implementazione per *IoT* di uno algoritmo *SHA-3* "sponge-based" [26][27]) usata nella sua implementazione CCurl, che si basa sull'uso della CPU del sistema che invia la transazione per calcolare una hash. Una volta costruito il *bundle* il dispositivo esegue quindi il puzzle fornendo a CCurl in input il *transactionObjectTrytes* (l'oggetto della transazione codificato in trytes) e un valore detto *Minimum Weight Magnitude (MWM)*, che rappresenta la difficoltá impostata del PoW. Il risultato del calcolo é una nonce che viene unito al *transactionObjectTrytes*, convertito in trits, e di nuovo fornito in ingresso alla funzione Curl per la validazione. Se il numero di zeri finali é uguale o superiore al valore *MWM*, allora la nounce é valida.

Questo algoritmo, nell'implementazione reale, puó essere eseguito direttamente sul dispositivo *IoT*, non richiedendo eccessive risorse, oppure delegato a server esterni che restituiscono semplicemente il risultato che viene poi inviato al *Tangle* insieme alla transazione. In questo secondo caso il funzionamento é quello riportato in Figura 2.24, dove il servizio intermedio effettua un "PoW-as-a-service" e poi invia il bundle alla *IOTA* network, in maniera trasparente rispetto al dispositivo *IoT*. La libreria che viene usata per ge-

stire l'invio delle transazioni tramite *Javascript* espone di base due possibili provider per questa operazione:

- thetangle (https://nodes.thetangle.org:443): scelto di default;
- powsrv (https://api.powsrv.io:443).

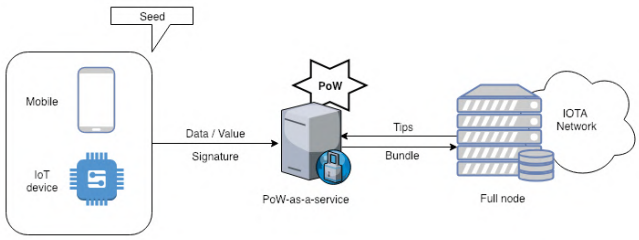


Figura 2.24: Invio di una transazione a IOTA usando un PoW-as-a-service [28]

2.4.7 IOTA come DLT permissionless e permissioned

L'ecosistema di *IOTA* é composto da tre network, cioé tre *Tangle*:

1. Mainnet: network principale di *IOTA*, su cui la criptovaluta *IOTA* ha effettivo valore e viene scambiata;
2. Devnet: netwkork secondario mantenuto per essere usato dagli sviluppatori. Su questa rete la criptovaluta ha valore fittizio e puó essere generata arbitrariamente;
3. Comnet: altro network con le stesse caratteristiche della Devnet.

Mentre le prime due sono mantenute dalla *IOTA Foundation* la terza rete é gestita interamente dalla community. Tutte e tre le reti sono configurate in maniera da essere del tutto *permissionless*, conferendo a chiunque la possibilitá di configurare un proprio nodo e parteciparvi. Per configurare un nodo sono disponibili due implementazioni:

- Nodo IRI: prima implementazione basata su una *Java Virtual Machine (JVM)*;

- Nodo Hornet: nuova implementazione, che compila direttamente codice nativo e aumenta le prestazioni, rendendo possibile l'installazione anche su dispositivi embedded.

Ad oggi i nodi Hornet sono i piú utilizzati e si collegano alla rete scelta, tra le tre definite precedentemente, ricevendo transazioni e registrandole nel ledger.

La presenza dei nodi Hornet permette tuttavia anche di creare delle configurazioni personalizzate di tipo *permissioned*, realizzando dei veri e propri cluster di nodi in grado di gestire *Tangle* privati. Questo tipo di configurazione non é stata ad oggi ancora perfezionata e presenta una serie di problematiche che verranno analizzate nei capitoli successivi.

2.4.8 Account e indirizzi IOTA

Poiché *IOTA* permette agli utenti di scambiare token e informazioni tramite la *DLT* é necessario prevedere un sistema di identificazione della proprietá di un determinato asset: questa viene garantita tramite gli "account".

Nella logica di *IOTA* non sono presenti chiavi pubbliche e private ma si parla di *seed* e *address* [29]: il primo costituisce il sostituto per la chiave privata e non deve mai essere comunicato a soggetti terzi, in quanto abilita lo scambio di token per l'account corrente, mentre il secondo invece rappresenta una specie di portafoglio al quale possono essere inviati token o informazioni; per ogni *seed* é possibile avere svariati *address* (Figura 2.25), ognuno dei quali contiene un determinato quantitativo di $IOTA$.

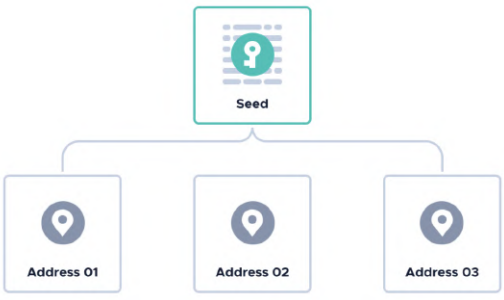


Figura 2.25: Rapporto tra seed e address in IOTA [29]

Tramite il *seed* é possibile garantire la proprietá di un determinato *address* e dei token o delle informazioni ivi contenute; la struttura é una stringa

di 81 trytes, che diventano 90 aggiugendo un checksum, il che porta ad un numero sostanzialmente illimitato di *seed* generabili pari a $8.7 \cdot 10^{115}$.
L'*address* puó essere comparato alla metá pubblica di una coppia di chiavi, e rappresenta un account personale che appartiene ad un determinato *seed*; ogni *address* é quindi collegato ad una chiave privata che permette di dimostrarne la proprietá. Anche un *address* viene rappresentato mediante una stringa di 90 trytes e puó essere tranquillamente condiviso in quanto solo il proprietario del *seed* che l'ha generato possiede la sua chiave privata e quindi puó dimostrarne la proprietá.

Nell'ecosistema di *IOTA* esistono due tipologie di *address*:

- One-time address;
- Merkle root address.

I primi sono detti "one-time" poiché utilizzano una "one-time signature" per cui, quando vengono usati per prelevare dei token, viene rivelata una parte della chiave privata: questo li rende estremamente sicuri fintanto che vengono effettuati dei depositi ma costringe a non usarli per depositare dopo il primo prelievo, marcandoli come "spesi". Un altro deposito potrebbe infatti essere rubato tramite un attacco a forza bruta sulla chiave privata.
Un *Merkle root address* é invece un indirizzo che utilizza uno schema di firma di tipo "Merkle", per cui vengono messi a disposizione un certo numero di bundle (gruppi di transazioni) che possono essere firmati in maniera sicura per confermare la proprietá di un determinato *address*. Questo tipo di indirizzi viene usato in quelle situazioni in cui é necessario mostrare la proprietá dello stesso *address* in piú bundle: il *Coordinator* per esempio utilizza questo tipo di *address* per firmare le *milestones*.

Da ogni *seed* é possibile ottenere 9^{57} *address*, ognuno dei quali é identificato da un "index" univoco e un "security level" che va da uno a tre: la tripletta *seed*, *index*, *security level* da come risultato sempre lo stesso *address*. Per provare la proprietá di un *address* l'utente firma la hash di un *bundle* con la chiave privata corrispondente in modo che altri utenti possano in futuro verificarne la proprietá.

Tutti gli indirizzi di tipo one-time sono generati usando una funzione chiamata "Kerl", versione ternaria dell'algoritmo di hashing "Keccak-384", facente parte della stessa famiglia di primitive crittografiche di SHA-3 [26],

che prende come input 48 bytes e restituisce 243 trit come output. L'algoritmo accetta come input *seed* e *index*, li converte in trits e li combina e ne calcola la hash per generare un "subseed" di 243 trits. Questo viene assorbito dalla *sponge function* e restituito 27 volte per ogni *security level*: il risultato é una chiave privata di lunghezza variabile (in base al *security level*). Ogni gruppo di 27 segmenti su cui é stato fatto la hash viene chiamato "key-fragment": una chiave privata possiede quindi un *key-fragment* per ogni *security-level*.

Per generare un *address* la chiave privata viene suddivisa in segmenti di 81 trytes, ognuno dei quali viene sottoposto ad hashing per 26 volte. Ogni *key-fragment* viene quindi sottoposto ad hashing una volta per generare un *digest* per ogni livello di sicurezza; questi *digest* sono poi combinati tra loro e ne viene fatto ancora una volta la hash per generare l'*address* di 81-tryte (Figura 2.26).

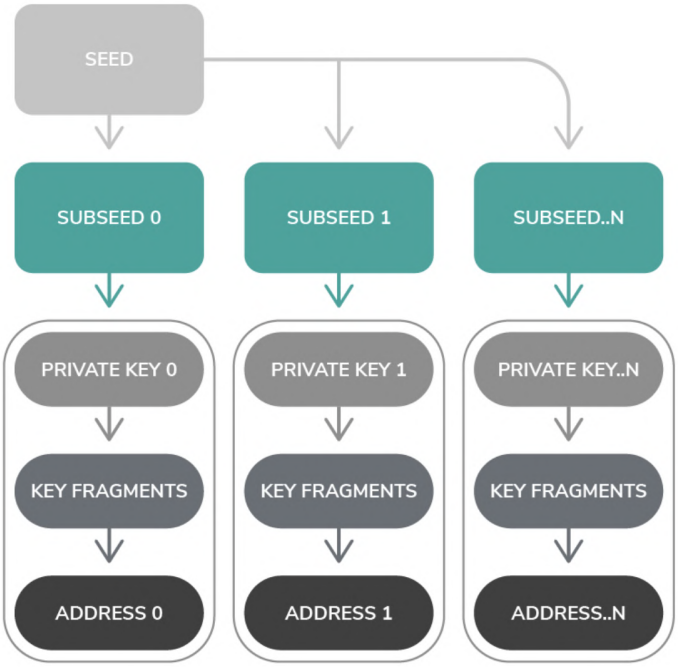


Figura 2.26: Processo di generazione di un address one-time [30]

La generazione degli *address* di tipo *Mekle* parte anch'essa dal processo sopra descritto ma in questo caso ne vengono generati n per costruire un *Merkle tree*, la cui *root* é il *Merkle address*. In base alla profonditá desiderata per l'albero vengono generati 2^{depth} *address*, che corrispondono al numero

di bundle che possono essere firmati per provare la proprietá del *Merkle root address*. Ai fini di questo elaborato i *Merkle address* non sono di ulteriore interesse, e non sono stati pertanto approfonditi i relativi meccanismi di verifica e generazione.

2.4.9 Tipi di transazioni e firma

All'interno del *Tangle* é possibile inserire due tipologie di transazioni:

- Zero-value;
- Value.

Indipendentemente dal tipo di transazione, solitamente queste vengono raggruppate all'interno di un bundle e numerate con indici crescenti.

Le transazioni a valore nullo sono quelle che possono contenere dati e informazioni e sono quindi quelle di maggiore interesse ai nostri fini; é possibile inserire i dati all'interno di due campi: `signatureMessageFragment` oppure `tag`. Per questo tipo di transazioni i nodi, non essendo presente scambio di valuta, effettuano semplicemente il *PoW* e controllano il timestamp, assicurandosi che non sia piú vecchio di dieci minuti rispetto al tempo corrente calcolato dal nodo stesso.

Per le transazioni con valore la situazione é diversa e vengono effettuati tre ulteriori controlli: il primo é quello sul valore della transazione, che non deve superare il numero massimo di token disponibili sulla rete; il secondo consiste nel check sull'effettiva proprietá, da parte del mittente, dei token scambiati; il terzo é un controllo sulla firma, per verificare se é valida. Quest'ultimo controllo viene effettuato solo sulle transazioni che effettuano un prelievo da un *address* e si effettua sfruttando il campo `signatureMessageFragment`.

Per realizzare una firma il proprietario dell'*address* da cui si sta effettuando il prelievo firma con la chiave privata corrispondente la hash del bundle e, nel caso questo valore sia troppo lungo per essere contenuto all'interno di una singola transazione, viene suddiviso nei campi `signatureMessageFragment` delle varie transazioni del bundle. L'algoritmo usato é di tipo *Winternitz one-time signature (W-OTS)* [31], la cui caratteristica principale é quella di essere "quantum-safe", nel senso in cui si ritiene che il tempo necessario per rompere l'algoritmo a forza bruta con un computer quantistico sia paragonabile a quello necessario a farlo con un computer tradizionale. Questa

caratteristica é peró accompagnata al fatto che lo schema rivela una percentuale ignota della chiave privata utilizzata per la firma. Il risultato, come giá accennato, é il fatto che sia sicuro prelevare da un *address* soltanto una volta, etichettandolo poi come "spent". Poiché l'interesse principale di questo elaborato é quello di trasferire dati e non valuta, verranno utilizzate in seguito soltanto le transazioni *zero-value* e non é stato pertanto approfondito il meccanismo di firma e di verifica delle transazioni con valore.

2.4.10 Considerazioni finali su IOTA

Il protocollo *IOTA* si é rivelato una soluzione particolarmente promettente per l'integrazione dei dispositivi *IoT*, sia del punto di vista teorico che poi da quello pratico. La sua capacitá di coinvolgere i dispositivi stessi nel mantenimento del *Tangle*, eliminando cosí le operazioni di mining e le relative commissioni, rende *IOTA* la soluzione probabilmente migliore per gli scopi di questo elaborato, almeno per quanto riguarda appunto la parte di gestione della sensoristica. Per quanto non sia una *blockchain* in senso stretto, il *Tangle* presenta le stesse caratteristiche di trasparenza, decentralizzazione e sicurezza, anche se su questi ultimi due punti é ancora necessario del lavoro da parte di *IOTA Foundation* per eliminare il *Coordinator* e raggiungere la completa decentralizzazione.

Un problema molto importante dell'ecosistema *IOTA* é la presenza dei cosiddetti "snapshot", eventi periodici in cui il *Tangle* della *Mainnet* (e anche quello della *Devnet* piú di frequente) vengono svuotati da tutte le transazioni vecchie, per poi ripartire da zero, mantenendo ovviamente intatti tutti i bilanci dei vari account.
Questa scelta, effettuata per motivi di spazio di archiviazione, é in netto contrasto con la caratteristica di immutabilitá e persistenza delle informazioni sulla *DLT*, fondamentale per la maggior parte dei modelli di business attuali. Sebbene *IOTA* preveda l'archiviazione delle precedenti istanze del *Tangle*, fornendo a chiunque li volesse i vari snapshot effettuati, il meccanismo di recupero di transazioni giá archiviate risulta ancora relativamente macchinoso e in contrasto con il concetto di semplicitá e completa trasparenza richiesto.

La soluzione ad oggi presente é chiamata *Chronicle*, e consiste in un plugin per un qualsiasi nodo *Hornet* che mantiene tutte le transazioni senza effettuare snapshot, il quale puó essere implementato in una logica permissioned, collegandolo quindi alla *Mainnet*. Le implicazioni di questa soluzione saranno analizzate nella pratica nei prossimi capitoli.

Capitolo 3

BlockWEEE: Implementazione di una blockchain of things per il tracciamento lungo la supply chain

In questo capitolo verranno presentati i risultati ottenuti a seguito della partecipazione al progetto Climate KIC dal titolo "Introducing blockchain technology in professional WEEE management (BlockWEEE)". Scopo ultimo del progetto era quello di valutare benefici e criticitá derivanti dall'uso della tecnologia *blockchain* nel contesto di un sistema di tracciamento dei rifiuti da apparecchiature elettriche ed elettroniche (*RAEE*) di tipo professionale, sfruttandone le potenzialitá per tracciare il prodotto lungo il suo "life-cycle" fino all'"end-of-life". Piú in generale lo scopo di questo elaborato é di proporre un soluzione relativamente generica che possa applicarsi ad una serie di casi studio che esulino anche potenzialmente da quello inizialmente proposto da *BlockWEEE*, come quello del monitoraggio ambientale o della tracciabilitá agroalimentare.

Per prima cosa verranno quindi definiti i requisiti identificati per realizzare questa tipologia di sistema, che si qualifica a tutti gli effetti come $BCoT$; successivamente sará presentato il test-bed sviluppato all'interno dei laboratori dell'Universitá di Bologna, mostrando l'architettura e fornendo una overview dei risultati. Queste considerazioni saranno corredate da test sulle prestazioni del sistema, effettuati nell'arco di diverse settimane di utilizzo; tali test hanno portato alla definizione di specifiche del sistema che potrebbero essere modificate in implementazioni future, per migliorarne prestazioni, scalabilitá e implementabilitá effettiva. Sono state inoltre evidenziate alcune criticitá ad oggi ancora da investigare nello specifico.

Le considerazioni fatte e le ricerche preliminari hanno evidenziato come non vi siano in letteratura esempi di sistemi completi giá funzionanti e utilizzati per lo scopo del progetto, cioé il tracciamento dei RAEE professionali; per questo motivo verranno in un primo momento presentate le specifiche richieste per questo utilizzo, cercando poi di comprendere se i casi presenti in letteratura siano adattabili al progetto in questione. Sono infatti giá presenti casi studio per differenti campi di applicazione, come quello agroalimentare e logistico. Partendo da questi esempi si é cercato di definire una proposta di sistema generico per poi scendere nello specifico esponendo una soluzione per il caso di tracciamento di un frigorifero ad uso industriale.

A livello operativo lo scopo di *BlockWEEE* era quello di studiare la fattibilitá ed eventualmente realizzare un prototipo di sistema che potesse essere utilizzato come punto di partenza al fine di fornire, in ultima istanza, una proposta concreta a differenti attori interessati al progetto. Per svolgere questo elaborato quindi si é deciso di sviluppare un sistema relativamente *general-purpose*, che sfrutti il concetto di $BCoT$ per garantire le qualitá comuni ricercate dai sistemi *blockchain* e IoT contemporaneamente, quali: trasparenza, affidabilitá, decentralizzazione del controllo, sicurezza e scalabilitá. Il prototipo risultante é stato quindi sviluppato a partire dalle richieste e dai benefici che i vari stakeholders di *BlockWEEE* si aspettano da questo tipo di sistema, senza peró che questa caratteristica sia limitante e prendendosi la libertá di analizzare ulteriori casi studio. Il tutto é stato realizzato perseguendo lo scopo ultimo di questa tipologia di sistemi, cioé promuovere una gestione consapevole e virtuosa dell'informazione, garantendo contemporaneamente la trasparenza e la tracciabiliá e fornendo informazioni chiare e certificate.

3.1 Caso studio e dei requisiti tecnici

3.1.1 Analisi delle specifiche del progetto BlockWEEE

Il problema del non corretto smaltimento dei *RAEE*(in inglese *WEEE*, da cui il nome del progetto) é per l'Europa ad oggi di grande importanza [32], poiché questi costituiscono uno dei maggiori flussi di rifiuti, con una percentuale di crescita del $3-4\%$ annuo, circa tre volte maggiore rispetto alle altre tipologie [33]. La gestione del flusso di rifiuti e del loro riciclo rappresenta un problema concreto per i sistemi attualmente in uso, a causa dell'eterogeneitá degli apparecchi che raggiungono il fine vita e della loro quantitá. Inoltre, al contrario di molte altre categorie di prodotti, i rifiuti elettronici ed elettrici spesso sono composti da sotto-parti che richiedono uno smaltimento specifico, al fine di non esporre operatori e ambiente a rischi derivanti dalle sostanze chimiche che potrebbero contenere e che potrebbero rilasciarsi a causa di una scorretta gestione.
Per questo motivo lo smaltimento dei *WEEE* richiede una gestione separata da quella degli altri flussi, insieme a sistemi di trasporto e logistica specifici e centri di disassemblaggio e gestione del fine vita. I dati raccolti a livello Italiano dal centro di coordinamento *RAEE* nel 2015 mostrano come, delle $165,000$ tonnellate di apparecchiature elettriche ed elettroniche (*AEE*) immesse nel mercato, solo il 42% (circa $70,000$ tonnellate) sia stato effettivamente trattato [34]. Questo significa che, supponendo che ogni anno la quantitá di *AEE* che viene dismessa sia pari a quella immessa nel mercato, mediamente una percentuale pari al 58% dei *RAEE* prodotti non viene dichiarato e quindi non viene smaltito correttamente.

La motivazione principale alla base di questa mancanza é da ricercare nel fatto che il processo di realizzazione delle *AEE* é spesso molto lungo, e la supply-chain ha spesso nodi al di fuori dei confini nazionali, a partire dall'estrazione delle materie prime necessarie a produrre i componenti, spesso effettuata in paesi esterni all'Unione Europea; si passa poi all'assemblaggio ed alle fasi di distribuzione e controllo, che vengono solitamente delegate dalle aziende ad attori secondari che si occupano singolarmente dell'una e dell'altra e non hanno una visione di insieme del procedimento. Questo modello di lavoro, che tende a delegare compiti e responsabilitá ad attori terzi, allungando e aggiungendo complessitá alla supply-chain, é ad oggi consolidato sopratutto per le grandi multinazionali; il risultato sono supply-chain frammentate e una limitata possibilitá di tracciare le informazioni e recuperare i dati trasversalmente. Il progetto *BlockWEEE* ha espressamente lo scopo di ricercare nella *blockchain* l'opportunitá di gestire sistemi di tracciamento trasparenti,

sicuri e affidabili, che siano distribuiti geograficamente tra attori differenti e che garantiscano un controllo trasversale sull'informazione. Il sistema che dovrá essere delineato deve garantire la possibilitá di creare, integrare e infine recuperare il percorso di vita completo dell'oggetto lungo la supply-chain.

3.1.2 Da IoT e blockchain alla BCoT

Ad oggi le tecnologie *IoT* sono fortemente inserite all'interno delle supply-chain industriali, dove vengono utilizzate per automatizzare una parte importante dei processi, aumentandone l'efficienza e la qualitá. I principali vantaggi che questo tipo di tecnologie possono fornire sono:

- Decentralizzazione del controllo;
- Grande varietá di dispositivi tra cui scegliere;
- Elevate interoperabilitá tra dispositivi diversi, e quindi grande possibilitá di integrazione dei sistemi;
- Eterogeneitá dei dati che possono essere raccolti e manipolati.

Rimane tuttavia di grande importanza la questione della sicurezza, poiché questo tipo di dispositivi spesso elabora e gestisce informazioni critiche, che devono essere utilizzate in maniera corretta e sopratutto devono essere garantire e certificate mediante un qualche strumento.

In questa ottica la *blockchain* si pone come strumento di vera e propria certificazione dell'informazione, permettendoci di eliminare l'ultimo tassello "umano" e centralizzato rimasto all'interno dell'ecosistema, quello del controllo e della validazione dei dati, sia in operazioni intermedie che per stoccaggio finale. La caratteristica della *blockchain* di essere immutabile e decentralizzata risponde almeno in parte al problema di sicurezza dell'*IoT*, garantendo la possibilitá di conservare i dati direttamente all'interno di un *ledger* dove questi non possano essere alterati. Rimane tuttavia il problema del collegamento del dispositivo embedded con la *blockchain*, poiché le prime implementazioni di questa tecnologia non presentano caratteristiche compatibili con una integrazione diretta nell'ottica dei sistemi embedded..

La prerogativa dell'*IoT* é infatti quella di essere composto da dispositivi embedded di piccole dimensioni e il cui consumo energetico sia quasi trascurabile rispetto al consumo totale del sistema in cui sono inseriti; non é

pertanto possibile utilizzare su dispositivi *IoT* i tradizionali strumenti crittografici usati dalle *blockchain*, i quali richiedono uno sforzo computazionale non sostenibile in termini di CPU. Per ovviare a questo problema sono state ricercate implementazioni *blockchain* che siano in grado di ridurre i consumi (ad esempio *IOTA*) oppure sono state pensate particolari implementazioni topologiche che permettano di raggruppare il consumo energetico e la potenza della CPU su un singolo nodo a cui facciano capo una serie di dispositivi *IoT* distribuiti su un'area estesa. D'altro canto la tecnologia *blockahin* presenta una serie di caratteristiche che motivano lo sforzo di ricerca atto ad abilitarne la commistione con l'*IoT*, tra le quali spicca la possibilitá di fornire i dati, certificati e garantiti dal sistema, contemporaneamente sia al produttore che all'utente finale (o consumatore in caso studio specifici), migliorando la tracciabilitá degli oggetti e l'affidabilitá delle informazioni lungo la supply-chain.

La parola chiave per comprendere l'urgenza e l'importanza della *BCoT* é "tracciabilitá", e rappresenta la possibilitá di registrare storia, posizione, distribuzione di un prodotto o di una parte di questo, mantenendo contemporaneamente una marcatura temporale di ogni operazione eseguita sullo stesso, il tutto tramite un identificativo univoco (ID), che viene registrato sulla *blockchain*. La realizzazione del paradigma *BCoT* consiste quindi nel creare oggetti intelligenti, collegati gli uni gli altri grazie alla rete internet, che siano in grado di eseguire il sensing dell'ambiente circostante e raccogliere informazioni. I dispositivi possono poi anche essere utilizzati per effettuare monitoraggio costante dell'ambiente o di determinati sistemi, sempre nell'ottica di lasciarli agire e comunicare in maniera indipendente tra loro e con l'ambiente esterno, con il minimo intervento umano. Oltre che sensori, i dispositivi *BCoT* possono anche essere attuatori, nel senso in cui essi sono in grado di agire autonomamente sull'ambiente in seguito agli stimoli ricevuti; i dati cosí utilizzati o registrati vengono poi inseriti automaticamente all'interno di una *blockchain* per essere disponibili, certificati e trasparenti.
La *BCoT* é risponde cosí alle richieste dell'industria moderna, le quali provengono sia dalle aziende stesse, interessate a maggiore controllo e supporto logistico, sia dai consumatori e dai clienti finali che possono usufruire di questi dati per aumentare la loro fiducia nell'azienda. L'automatizzazione delle operazioni mediante *BCoT* permette inoltre di aumentare drasticamente l'efficienza dei processi produttivi, eliminando il piú possibile la componente umana all'interno dei processi e l'intervento di autoritá terza all'interno della supply-chain, assicurando contemporaneamente la trasparenza dei processi.

3.1.3 Definizione dei requisiti di un sistema BCoT

Come anticipato, per potersi definire tale un sistema *BCoT* deve rispettare una serie di requisiti fondamentali che garantiscano un sufficiente livello di trasparenza e tracciabilitá dell'informazione, mantenendo i costi relativamente contenuti e le risorse da impiegare, in termini energetici e di sforzo umano per manutenzione e gestione, al minimo possibile.

Primo tra tutti il sistema deve essere stabile e robusto, nel senso in cui deve garantire un alto livello "disaster recovery" e "faul-tolerance". Questo é necessario in quanto l'eliminazione del contributo umano e centralizzato all'interno del sistema lascia spazio ad una serie di errori che devono essere gestiti in maniera automatica senza che il sistema sia passibile di malfunzionamenti. Inoltre, parlando di dispositivi che devono sempre essere collegati a fonti di alimentazione e alla rete internet (almeno in linea generale), é necessario prevedere una serie di strumenti atti a contenere eventuali malfunzionamenti esterni al sistema che compromettano una delle due tipologie di rete.
Il concetto di stabilitá del sistema deve riflettersi anche sull'informazione ivi contenuta, la quale deve essere garantita immutabile, perenne e sempre accessibile. I sistemi *blockchain* scelti per abilitare la *BCoT* devono quindi garantire resistenza a due problemi che affliggono questa tecnologia:

- Attacco del 51%;
- Bizantine fault.

In generale infatti un problema della *blockchain* é quello per cui un utente malevolo, che cerchi di modificare per proprio tornaconto le informazioni, sia in grado di usare la propria potenza computazionale per alterarle a proprio favore. Nei sistemi *blockchain* tradizionali, qualora un utente possedesse oltre il 51% della potenza computazionale della rete, sarebbe per lui altamente probabile riuscire nel proprio intento. Per ovviare a questo, che risulta un problema patologico per come la *blockchain* stessa é stata pensata, e presente anche in implementazioni di *DLT* come il *Tangle* di *IOTA*, l'unica soluzione é utilizzare reti abbastanza mature per cui le probabilitá per un utente di possedere il 51% della potenza computazionale totale siano molto basse. Questo problema per esempio é superato in *Bitcoin* ma ancora del tutto attuale nel caso di *IOTA*.
Per quanto riguarda la *BFT* i sistemi che sono stati analizzati presentano tutti questa caratteristica, inevitabile in ogni situazione in cui si ponga il problema del consenso distribuito, e resistono quindi ad una serie di errori,

involontari o indotti da comportamenti fraudolenti, che si presentino al momento di raggiungere il consenso.
Un secondo requisito fondamentale per un sistema *BCoT* é quello di essere efficiente a livello energetico, nel senso in cui il consumo di energia, e quindi il costo diretto che ne segue, derivante dall'uso di questo, deve essere piccolo se paragonato al costo totale dell'ecosistema in cui viene inserito. Questa caratteristica, piú di ogni altra, é sufficiente ad escludere *Bitcoin*, in quanto il suo consumo stimato ammonta a $75TWh$ annui, il che é decisamente troppo elevato per essere sostenuto in ottica futuribile.
Altro problema fondamentale, e quindi terzo requisito dei sistemi *BCoT*, é la conservazione delle informazioni relative ai partecipanti al sistema, e nello specifico la conservazione delle chiavi private e di tutte quelle informazioni che abilitano la possibilitá di ricollegare in maniera univoca informazione e dispositivo che l'ha inserita nella *blockchain*. Un sistema *BCoT* deve prevedere un sistema efficace di conservazione di questo tipo di informazioni, in modo che siano protette da eventuali attacchi esterni. Conservare le chiavi e questo tipo di informazioni rappresenta un grosso problema e, ad oggi, un grande punto di domanda sull'efficienza di questo tipo di sistemi, poiché introduce un inevitabile "single-point of failure".
Ultimo problema é quello dell'occupazione di spazio di archiviazione, poiché il concetto alla base dell *blockchain*, quale l'immutabilitá, impone l'impossibilitá di eliminare informazioni e il conseguente aumento costante dello spazio occupato. Anche se ad oggi i sistemi presenti non presentano dimensioni insostenibili in termini di memoria é comunque da tenere in considerazione come un uso massivo della tecnologia possa portar presto il rateo di aumento di spazio di archiviazione occupato ad essere molto rapido. É quindi requisito ulteriore per un sistema *BCoT* utilizzare una implementazione di *blockchain* che sia adatta a sostenere l'alto numero di transazioni effettuate.

3.1.4 Le sfide dell'integrazione tra blockchain permissionless e IoT

Come analizzato nella sezione precedente la *BCoT* presenta una serie di requisiti che devono essere rispettati affinché il sistema sia implementabile. Questi, tuttavia, non sono sufficienti per definire il sistema funzionante e sopratutto funzionale, in quanto i primi esempi di integrazione tra *blockchain* e *IoT* hanno portato alla luce alcune inefficienze e conseguenti problemi derivanti dall'attuale implementazione della tecnologia *blockchain* di tipo *permissionless*.

Il principale problema in questo tipo di integrazione é dato dall'alto tasso di transazioni per secondo (TPS) generate da un numero massivo di dispositivi *IoT*, il quale risulta incompatibile con la necessitá dei *ledger permissionless* di evitare spam e quindi limitare il numero di transazioni effettuabili contemporaneamente. A questo si aggiunge la presenza di tempi di conferma relativamente lunghi nelle tradizionali implementazioni e la richiesta di commissioni di trasferimento (come nel caso di *Bitcoin*) che non facilitano l'integrazione con dispositivi il cui scopo sia quello di inviare anche varie transazioni al minuto per effettuare monitoraggio. Per risolvere questo tipo di problematiche sono stati proposti protocolli *DLT* alternativi che non sfruttano il *ledger blockchain* tradizionale o che lo implementano in maniera *permissioned*, in modo da limitare la necessitá di algoritmi di consenso onerosi ed eliminare del tutto le commissioni.
Una di queste implementazioni é sicuramente *Hyperledger*, che rappresenta ad oggi il piú sviluppato esempio di *blockchain permissioned*, accanto alla quale troviamo *IOTA*, pensata giá dalla sua realizzazione iniziale per supportare micro-transazioni senza commissioni ed essere integrata su dispositivi *IoT*.

Risulta quindi evidente come ad oggi, per applicazioni di tipo industriale, siano piú appetibili soluzioni di tipo *permissioned*, che si dimostrino adatte a gestire alti volumi di dati in maniera contemporaneamente sicura e affidabile. Questo tipo di implementazione risponde certamente alla richiesta di una immutabilitá dell'informazione e di una sua decentralizzazione, abilitando una serie di modelli di business dove gruppi di aziende condividono dati internamente tra partner autorizzati; é quindi decisamente possibile ottenere "tracciabilitá" completa di una supply-chain tramite un sistema *BCoT* che sfrutti *blockchain permissioned*. Tuttavia, l'ecosistema che ne emerge non é considerabile del tutto trasparente, per quanto i prodotti siano tracciati; rimane infatti una criticitá di questo tipo di configurazioni il fattore "trasparenza", che deve essere sempre garantita nell'idea di creare un "trust-layer" decentralizzato sui dati che vengono forniti. Infatti, sebbene le *blockchain permissioned* garantiscano gli stessi benefici delle implementazioni *permissionless* ai loro utilizzatori, non garantiscono completamente l'immutabilitá e la trasparenza a tutti gli utenti esterni che debbano solo visualizzare i dati o parte di essi. In una implementazione *permissioned*, anche qualora si decida di fornire ad utenti esterni al sistema i soli permessi di lettura sui dati, é impossibile per l'azienda che li mostri garantire che questi siano integrali e che non vengano o verranno mai manipolati in qualche forma, se non ponendosi come autoritá superiore garante di queste qualitá; l'autoritá che gestisce il sistema *permissioned* puó infatti in ogni momento modificare

il livello di permesso per nascondere dati agli utenti, senza che questi siano a conoscenza dell'operazione, non facendo effettivamente parte del network. L'introduzione del concetto di fiducia centralizzata, derivante dall'introdurre un'autoritá garante dei dati, sopratutto qualora questa fosse la stessa che li fornisce (e che quindi non ha nessun interesse a mostrare dati che siano reali ma che potrebbero alimentare un'immagine negativa) é in netto contrasto con le caratteristiche della *blockchain* e ancora di piú della *BCoT*. Questa conseguenza delle *blockchain permissioned* é la principale fonte di discussioni in ambiente *blockchain*, dove si scontrano due differenti opinioni: da una parte i sostenitori dell'uso delle sole *blockchain* tradizionali (*Bitcoin*, *Ethereum* e simili), dall'altra coloro che vedono anche in queste nuove implementazioni *permissioned* la possibilitá di allargare il campo di applicazione di questa tecnologia.

Volendo provare a proporre una soluzione per risolvere questa controversia, sfruttando il caso industriale di partenza di *BlockWEEE*, si é deciso per un'implementazione di tipo ibrido che ad oggi non trova alcun riscontro in letteratura. Il sistema che verrá presentato, dove ad una *blockchain permissioned* realizzata tramite *Hyperledger Fabric* é affiancata la registrazione dei dati sul *Tangle* di *IOTA*, é pensato per unire benifici di *blockchain permissioned* e *permissionless*, mantenendo i dati sempre in duplice copia e garantendoli tramite un *ledger* decentralizzato e pubblico all'utente finale; mentre l'azienda utilizza la *blockchain* privata per velocizzare i processi di scambio informazioni e la loro registrazione, l'utente comune é in grado in ogni momento di controllare autonomamente la completezza e integritá dell'informazione ottenuta.

I nodi *BCoT* possono essere quindi usati per collezionare dati provenienti da tutti gli attori della catena di produzione facenti parte di un consorzio, utilizzando *Fabric* per realizzare la rete che colleghi produttori, operatori logistici e distributori di un prodotto. I nodi *blockchain*, installati in ogni punto chiave della supply-chain, realizzano un DB decentralizzato e condiviso sul quale i vari attori possono registrare dati con la sicurezza che questi non possano essere modificati o eliminati da parte di altri utenti; sullo stesso DB poi i dispositivi *IoT*, usati per automatizzare il maggior numero possibile di processi, agendo laddove l'intervento umano potrebbe portare ad una potenziale alterazione o falsificazione dell'informazione: in questo modo é possibile ottenere un sistema che garantisca trasparenza e affidabilitá su tutta la supply-chain, fornendone un'immagine trasversale a tutti i vari attori che vi partecipano, oltre che ad utenti esterni.

3.1.5 Alcuni casi presenti in letteratura

Come giá accennato non sono presenti in letteratura esempi pratici di sistemi atti al tracciamento di *RAEE* professionali tramite *BCoT*; sono tuttavia numerosi i progetti, alcuni piú fortunati di altri a livello implementativo, in campo agrifood e in campo logistico, che utilizzano la *blockchain* integrata con sensoristica e altri dispositivi embedded per fornire controllo orizzontale e trasparenza sulla supply-chain. É da notare come due terzi dei progetti presentati siano stati realizzati in collaborazione con IBM, e quindi utilizzando il framework *Hyperledger*: questo motiva ancora di piú l'interesse sullo studio di questa tecnologia, poiché sono giá presenti casi da cui trarre dati e valutazioni di fattibilitá.

A seguire verranno presentati tre di questi progetti, mostrandone brevemente le motivazioni e i risultati, oltre che il sistema utilizzato.

Walmart. Walmart, in collaborazione con IBM, ha proposto un sistema di tracciamento [36], basato su *BCoT*, per fornire ai propri clienti un portfolio digitale contenente tutte le informazioni riguardanti origine, materie prime e percorso di alcuni prodotti venduti. Tramite *Hyperledger* si é risolto il problema della frammentazione delle informazioni relative alle varie supply-chain dei prodotti alimentari, registrando record diversi per ogni fase della produzione e mantenendo informazione completa sul ciclo di vita del prodotto "from -farm-to-fork". Il principale beneficio riportato da Walmart consiste nell'aumento dell'efficienza della supply-chain e nell'aumento della fiducia del consumatore riguardo alla catena; grazie al sistema infatti il consumatore puó seguire facilmente il percorso del prodotto al momento dell'acquisto. Un secondo beneficio riscontrato é quello della diminuzione del costo di inventario, che viene redatto dal sistema in maniera automatica. Ad oggi Walmart, dopo aver testato la tecnologia per la supply-chain del mango, la utilizza per il tracking di oltre trenta prodotti.

Provenance. L'azienda Provenance [37] ha proposto un sistema completo di tracciamento del tonno pescato in Indonesia, realizzato integrando *blockchain* con smart tags e dati provenienti da applicazioni mobile. Tramite questo sistema il consumatore é in grado, tramite una semplice applicazione mobile, di ricostruire il percorso del pesce, fino a conoscere esattamente quale produttore l'ha pescato e quando, oltre a tutti gli step di processo che l'hanno portato fino al momento dell'acquisto.

Maerks. Sfruttando anche in questo caso una collaborazione con IBM

l'azienda Maersk [38], principale operatrice di spedizioni tramite container al mondo, ha sviluppato un sistema che integra *IoT* e *blockchain* per registrare temperatura, posizione e condizioni generali dei container nelle varie spedizioni, salvando i dati automaticamente all'interno del *ledger*. Questa implementazione ha fornito all'azienda un'enorme vantaggio per quanto riguarda il risparmio cartaceo, con una conseguente diminuzione dei costi stimata nel 15% del valore totale della spedizione. Altro punto a favore di questo tipo di sistema é stato riscontrato nella maggiore sicurezza dei trasporti (evitando frodi come bolle di accompagnamento duplicate o simili) e nella possibilitá di snellire i processi di controllo manuale, sfruttando i dati raccolti automaticamente e una serie di allarmi collegati ai container qualora i parametri dovessero risultare anomali.

3.2 Progettazione e implementazione del prototipo di nodo IoT

Lo scopo principale di questo elaborato é quello di presentare la soluzione prototipale che superi e implementi il *PoC* finora descritto, mostrando in maniera pratica la fattibilitá del sistema proposto e riportando una serie di dati di partenza utili per una futura implementazione.

Le esperienze analizzate precedentemente hanno permesso di costruire il punto di partenza del sistema di monitoraggio richiesto per il testbed *BlockWEEE*, realizzato tramite una rete di nodi *IoT* che comunicano con una struttura *blockchain*. Nello specifico, i nodi *IoT* sono stati realizzati fisicamente utilizzando una serie schede PCB contenenti sensori e integrabili through-hole, uniti tramite breadboards a dispositivi embedded low-cost, caratterizzati inizialmente tutti da un'interfaccia wireless verso la rete internet; i nodi di raccolta dati cosí realizzati si sono dimostrati in grado di campionare dati provenienti dall'ambiente circostante, in maniera autonoma o con logica "triggered" (quindi rispondendo al superamento da parte di un parametro di una soglia critica), e registrarli su un *ledger* generico, le cui specifiche saranno fornite nel seguito.

Gli studi e i test realizzati all'interno dei laboratori di telecomunicazioni dell'Universitá di Bologna, Campus di Cesena, hanno dimostrato la fattibilitá del suddetto sistema e la sostenibilitá, almeno parziale, dello stesso per gli scopi di questo progetto.

3.2.1 Progettazione della componente hardware

La selezione della componente hardware é stata il primo passo svolto per la definizione pratica del prototipo da realizzare, al fine di validare le assunzioni sulle potenzialitá del sistema finora fatte. Per prima cosa quindi si é cercato di valutare quali potessero essere le migliori implementazioni per un sistema il cui principale scopo era quello di essere:

- Affidabile;
- Relativamente a basso costo;
- Di semplice integrazione;
- In grado di garantire bassa latenza e alto TPS.

I Raspberry Pi, realizzati dalla Raspberry Pi Foundation, sono risultati ad una prima analisi la scelta migliore per gli scopi del progetto per varie motivazioni, tra le quali la principale é il fatto che siano micro-computer embedded su una scheda singola, quindi dispositivi in grado di sostenere un sistema operativo (SO) e fornire quindi tutti i vantaggi relativi. É infatti possibile utilizzare direttamente il dispositivo per scrivere il codice per la lettura dei dati, compilarlo e mandarlo in esecuzione, utilizzando anche diversi linguaggi, in base alle esigenze, senza doversi avvalere dell'uso di un computer esterno (come invece sarebbe stato necessario per altre tipologie di dispositivi embedded senza SO).
Questo tipo di dispositivo offre grande flessibilitá qualora sia necessario effettuare modifiche al codice in fase di prototipazione e test, oppure anche per eseguire interventi di manutenzione in fase finale, quando il dispositivo sia giá stato montato nel suo alloggio definitivo, senza dover intervenire con un computer portatile.

Sebbene siano a tutti gli effetti dei computer, questi dispositivi presentano anche alcune caratteristiche fondamentali dei dispositivi embedded tipici del mondo IoT, quali la disponibilitá di porte input/output GPIO e la possibilitá di programmare direttamente a livello hardware accedendo ai pin del chip interno. Questa caratteristica permette una connessione semplice e diretta di sensoristica e altri dispositivi e abilita la possibilitá di usare direttamente le schede di raccolta dati mediante librerie apposite.

La scelta di Raspberry Pi ha peró portato alla necessitá di individuare quale potesse essere il dispositivo migliore per gli scopi del progetto, poiché

sul mercato ne sono disponibili varie versioni. Si é deciso in un primo momento di testare tutte e quattro le principali schede fornite da Raspberry Pi Foundation, quindi: Raspberry Pi Zero, Raspberry Pi 2, Raspberry Pi 3 e Raspberry Pi 4. La differenza principale tra questi dispositivi sta nella quantitá di RAM fornita, nella potenza della CPU e nel costo. Aumentando l'indice che segue il nome del componente aumenta il costo e la potenza della CPU, mentre la RAM dipende dal modello specifico scelto e sono disponibili alcune opzioni. Per gli scopi del progetto sono stati usati principalmente (Figura 3.1):

- Raspberry Pi 4, 4GB RAM;
- Raspberry Pi Zero WH (con pin giá montati).

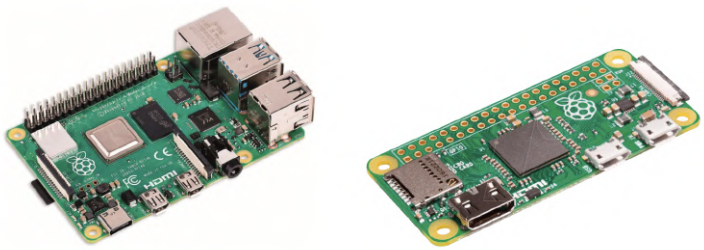


Figura 3.1: Da sinistra: Raspberry Pi 4 e Raspberry Pi Zero

Per quanto riguarda invece la componente di sensoristica (Figura 3.2) non é stato ritenuto di particolare importanza il tipo di sensori da integrare in questa prima fase, poiché ad oggi la maggiorparte di quelli disponibili viene venduto corredato da librerie apposite per integrazione plug-and-play. La scelta é ricaduta quindi su un cluster di sensori che potesse realisticamente implementare un primo controllo sull'ambiente circostante un oggetto: sono state scelte diverse schede di raccolta dati preassemblate su PCB, installabili mediante pin I/O direttamente sui Raspberry oppure utilizzando una breadboard come supporto fisico. Nello specifico:

- Sensore di temperatura e umiditá: Mikroe MIKROE-3469 11 Click Temp Hum, scheda integrata realizzata da Mikroe su cui é montato un HDC1080 di Texas Instruments, insieme alla circuiteria necessaria per il collegamento;
- Sensore di temperatura e umiditá: DHT11, sensore stand-alone su scheda PCB con pin per il montaggio through-hole;

- Giroscopio e accelerometro: Adafruit 6DOF MPU6050, scheda PCB di Adafruit che monta il sensore di TDK e la componentistica necessaria al collegamento.

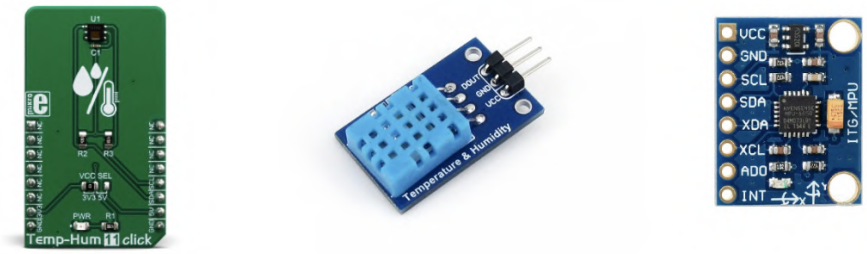


Figura 3.2: Da sinistra: MIKROE-3469 11 Click Temp Hum, DHT11, 6DOF MPU6050

La scelta é stata ridotta per soli motivi di costo, ma é in linea generale pensabile integrare coi nodi IoT qualsiasi sensore che possieda pin I/O.

É da sottolineare la motivazione della scelta di un doppio sensore di temperatura e umiditá, la quale é stata effettuata in corso d'opera, durante la realizzazione e i primi test del prototipo, equipaggiato inizialmente solo con MIKROE-3469. Infatti i primi test mostravano una mancanza di reattivitá da parte del sensore, che non risultava in grado di intercettare cambiamenti veloci della temperatura o dell'umiditá, come per esempio quelli dovuti al passaggio sopra di esso con un accendino. Empiricamente si é osservato come il valore convergesse a quello corretto solo dopo tempi approssimativamente pari a 30 secondi, valore non accettabile in future applicazioni in cui fosse necessario intercettare spike di temperatura e reagire tempestivamente. Per questo motivo si é scelto di affiancare al primo sensore un secondo, il DHT11, che non presenta queste problematiche. La motivazione del mantenimento del primo sta tuttavia nel fatto che, sebbene esso impieghi un tempo non trascurabile a convergere, questo fornisca misure con errore $\pm 0.2°$ (o $\pm 0.2\%$ per l'umiditá); al contrario il DHT11 fornisce misure con errore teorico pari a $\pm 0.5°$, anche se in sede di test questi valori sono risultati molto ottimistici, e si é arrivati anche a differenze di piú di un grado tra le due misure. La soluzione quindi é stata trovata a livello di codice: normalmente la misura considerata attendibile é quella piú precisa, cioé quella proveniente dal MIKROE-3469, ma, qualora il DHT11 individuasse situazioni sporadiche di

errore, con conseguenti spike dei dati letti, il MIKROE-3469 viene temporaneamente utilizzato per la sola correzione dell'incertezza e, per circa un minuto viene registrata misura ottenuta dall'DHT11 (dopo la correzione). In Figura 3.3 viene mostrata la scheda di raccolta dati montata su breadboard.

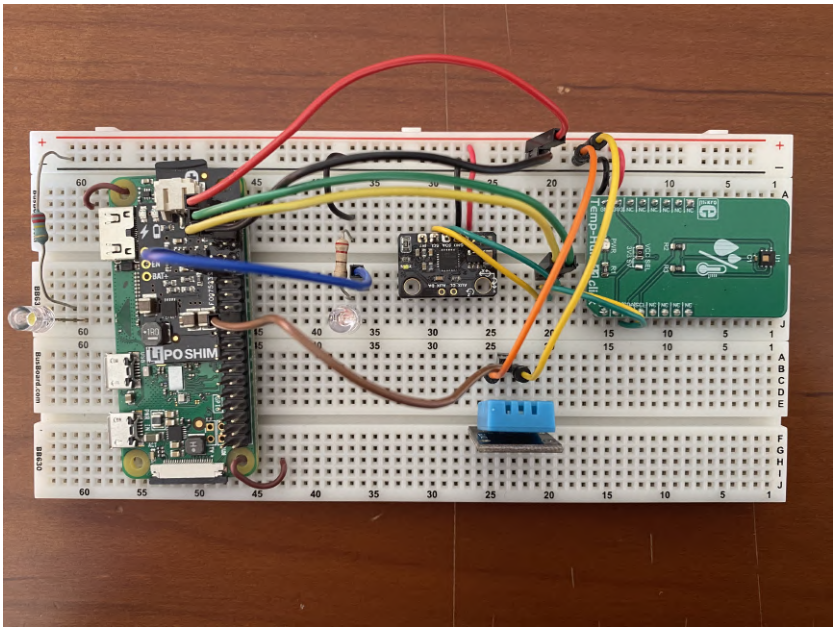


Figura 3.3: Scheda di raccolta dati basata su Raspberry Pi Zero

Per realizzare il sistema si é deciso di utilizzare contemporaneamente sia i Raspberry Pi Zero che i Rapsberry Pi 4, con la seguente logica: i primi funzionavano come stazioni di raccolta dati collegate direttamente alla sensoristica e comunicavano con il secondo (che funzionava come server), in logica $1 : N$ con topologia a stella; solo il Pi 4 era dotato di interfaccia verso la *blockchain* e vi comunicava i dati ricevuti, registrandoli definitivamente sul *ledger*. Questa scelta é stata fatta in fase di progettazione per due motivi principali:

1. Maggiore scalabilitá;
2. Minore costo in termini energetici ma anche in termini prettamente legati all'acquisto di hardware.

Il server centrale, infatti, elimina per i dispositivi sottostanti la complessitá computazionale di interfacciarsi alla blockchain, e sopratutto la necessitá

di essere sempre accesi e connessi ad internet. Inoltre, questa scelta rende possibile l'aggiunta di una nuova stazione di raccolta dati semplicemente configurando un nuovo Pi Zero e inserendovi l'indirizzo di rete del server, rendendo quindi il sistema altamente scalabile e personalizzabile anche una volta implementato effettivamente sul campo.
Dal punto di vista del risparmio energetico abbiamo poi la possibilitá di sostituire, in un secondo momento, i Pi Zero con dispositivi non dotati di connessione ad internet, prevedendo piuttosto un collegamento a radiofrequenza verso il server, salvaguardando il consumo energetico e abilitando la possibilitá di alimentare i nodi di raccolta dati tramite batteria o pannello fotovoltaico. Lo schema generale del sistema é riportato in Figura 3.4.

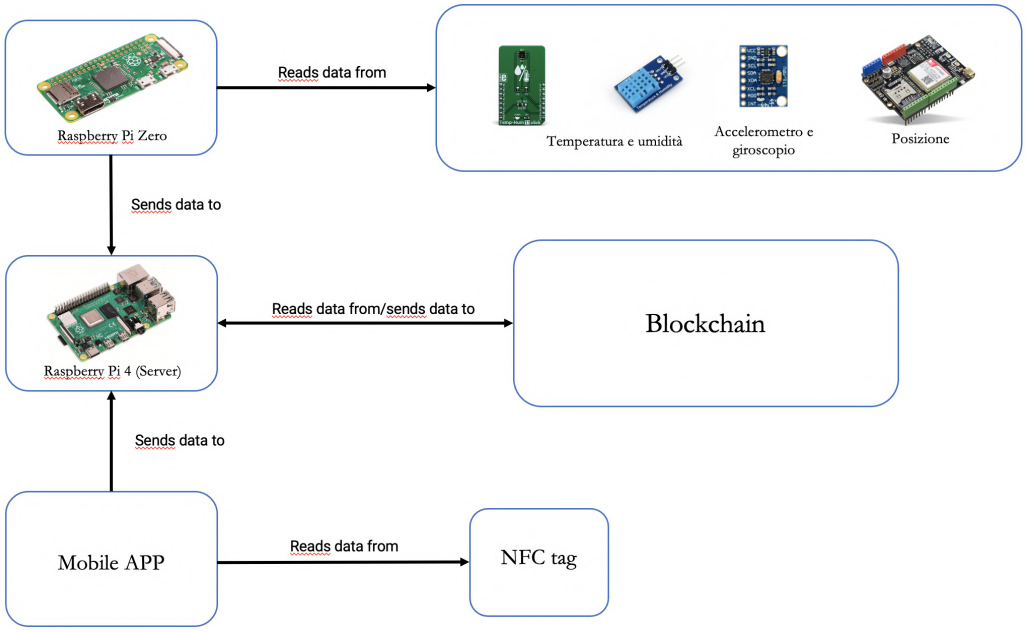


Figura 3.4: Diagramma generale dei componenti del sistema

L'alternativa a questa implementazione avrebbe dovuto prevedere l'integrazione, direttamente sui nodi *IoT*, della logica di scrittura su *blockchain*. Sebbene questo sia teoricamente fattibile, essendo disponibili per esempio librerie *Python* che permettono il collegamento direttamente con il *Tangle*, questo avrebbe previsto alcune criticitá:

- La scrittura su *blockchain* richiede necessariamente la connessione ad internet, la quale richiede solitamente che il dispositivo sia alimentato costantemente e non possa essere spento temporaneamente negli intervalli in cui non deve campionare dati. Questo, sebbene nel caso dei

Pi Zero non crei effettivi problemi, potrebbe in futuro creare criticitá nell'uso di dispositivi piú economici basati su microcontrollore (come per esempio Arduino o simili) e quindi senza SO; questi, infatti, non possiedono interfaccia nativa verso la rete internet e sono pensati per funzionare con l'interrupt del clock, accendendosi solo qualche secondo per campionare e inviare i dati, e ritornando poi in modalitá di risparmio energetico;

- La libreria *Javascript* di *IOTA*, di cui si parlerá piú avanti, non supporta applicazioni lato client ma solo lato server, che non sono implementabili sui dispositivi scelti il cui scopo é interfacciarsi con i GPIO.

Parallelamente ai nodi di raccolta dati é stata realizzata un'applicazione mobile, su dispositivo Android, in grado di collegarsi alla *blockchain* e recuperare i dati registrati dai sensori, sfruttando una serie di tag NFC. Infatti ogni nodo é stato associato in maniera univoca con un tag NFC in modo che i blocchi registrati sulla *blockchain* contenessero, tra i campi, anche questo ID e fosse possibile ricercare tutti i dati raccolti da ogni singolo nodo.

3.2.2 Implementazione della componente hardware

Le schede utilizzate per questo progetto dispongono di due tipologie differenti di collegamento: mentre quella su cui é montato il DHT11 é direttamente collegabile tramite pin I/O e i dati possono essere letti con segnale digitale direttamente sui pin, la scheda MIKROE-3469 e il sensore MPU6050 si interfacciano con il Pi Zero con un protocollo I2C e quindi questo bus di comunicazione deve essere configurato. Il protocollo in questione viene utilizzato per far comunicare diversi sensori, montati su schede PCB differenti, sullo stesso bus, assegnandogli indirizzi di scrittura e lettura specifica a 7 bit al momento del collegamento della periferica.
Essendo un protocollo master-slave, con possibilitá di avere differenti slave contemporaneamente, il Pi Zero (master) possiede quindi due soli pin per l'interfacciamento, denominati SDA e SCL (i quali sono rispettivamente il 3 e il 5), sui quali vengono collegati in parallelo entrambe le schede PCB (slave), lasciando al dispositivo il compito di assegnarvi gli indirizzi specifici per separare i dati inviati. I segnali SDA ed SCL, rispettivamente "serial data" e "serial clock", vengono usati per pilotare il bus dal Raspberry, con il ruolo di master, che controlla entrambi gli slave mediante i loro indirizzi; ad ogni dispositivo collegato viene assegnato un indirizzo a 7 bit.

Il master, qualora volesse iniziare la comunicazione con lo slave, attiva la linea portandola bassa e invia un bit di start (S) seguito dall'indirizzo dello slave con cui vuole comunicare, seguito da un bit che indica se si sta effettuando una operazione di lettura o scrittura. Se all'indirizzo scelto corrisponde uno slave questo prende il controllo della linea dati sul successivo impulso alto del SCL e la forza bassa per comunicare un ACK.

Nel caso del Raspberry il bus I2C deve essere abilitato dalle impostazioni, accedendo come:

```
sudo raspi-config
```

e scaricando poi i tools per la visualizzazione facilitata:

```
sudo apt-get install -y i2c-tools
```

Una volta riavviato il dispositivo é possibile visualizzare gli indirizzi dei dispositivi I2C collegati tramite il comando da terminale:

```
sudo i2cdetect -y 1
```

Per il campionamento dei sensori é stato scelto il linguaggio di programmazione *Python*, che risulta come linguaggio naturale per la comunicazione con le periferiche dei dispositivi della famiglia Raspberry, senza necessitá di aggiungere strumenti per la compilazione del linguaggio *C*. Per l'interfacciamento con il sensori sono state usate le librerie gratuite presenti in rete, con alcuni piccoli accorgimenti:

- É stata impostata una risoluzione di 14 bit per la temperatura e l'umiditá all'interno dei file di configurazione della libreria del MIKROE-3469. Il DHT11, che é molto economico, non presenta possibilitá di configurazione di alcun tipo sulla risoluzione;
- É stato spento l'heater intergrato sulla scheda MIKROE-3469 sfruttato per eliminare la condensa che potrebbe generarsi in seguito alla lettura dell'umiditá, ma che puó poi causare problemi alla lettura della temperatura. Lo spegnimento dell'heater obbliga a inserire un ritardo tra le due misure.

Una volta collegati tutti i sensori, é stato delineato il processo di campionamento e invio al server.
Il sistema hardware si muove tra tre stati che includono sia il campionamento

dei dati che l'invio degli stessi alla *blockchain*, ed é pensato per attendere la risposta di quest'ultima prima di procedere all'invio di un nuovo dato; in un'architettura a *super-loop*, tipica dei sistemi embedded, il nodo é pensato per muoversi in maniera sequenziale tra i vari stati, reagendo di conseguenza. Nello stato iniziale, che corrisponde a "sistema spento", il Raspberry rimane comunque acceso (non é previsto per questi dispositivi un meccanismo di *power saving*, motivazione per cui piú avanti verrá proposta una soluzione alternativa) mentre i sensori sono collegati ma non richiedono corrente in quanto non necessitano di campionare dati o scrivere sul bus. Per definire la periodicitá della lettura dei dati ci si é basati esclusivamente sulle specifiche tecniche dei componenti, non volendo inserire un delay all'interno del codice; per questo motivo con "after period" intendiamo il tempo necessario al sistema per accorgesi che il dato precedente é stato ricevuto e campionare il successivo nel seguente modo:

1. Viene campionato il valore dell'umiditá leggendo l'indirizzo corretto sul bus I2C;

2. Si attende un secondo e poi si campiona anche il valore di temperatura. Questo é necessario poiché la lettura dell'umiditá richiede uno spike di corrente, che puó surriscaldare il PCB a causa della presenza di due resistenze da $4.7k\Omega$;

3. Si effettua il campionamento di accelerometro e giroscopio.

Il pacchetto dati da inviare é stato costruito come un oggetto di tipo JSON, non presente nativamente in *Python* e che quindi viene simulato tramite un oggetto generico che poi il server sará in grado di convertire. La libreria usata per l'invio é *response*, che abilita l'invio dati via richieste HTTP bloccanti, facendo aspettare il super-loop fintanto che non si riceve una risposta dal server. Il server, nel nostro caso il Raspberry Pi 4, una volta ricevuto il pacchetto, lo invia con una chiamata HTTP alla *blockchain* (come questo avviene sará chiarito successivamente) e viene notificato al nodo l'esito dell'operazione; nel caso di operazione fallimentare il server raggiunge il timeout e manda una risposta di errore al nodo il quale, dopo averla ricevuta, esce dalla sua situazione di blocco ed effettua un nuovo ciclo. Per comprendere meglio le specifiche di funzionamento é stata definita una macchina a stati che potesse comprendere tutte le fasi del processo, da svolgersi iterativamente con il periodo preimpostato (Figura 3.5).

Come specificato in precedenza il valore standard che viene letto e inviato é sempre quello della scehda MIKROE-3469, mentre il valore del DHT11

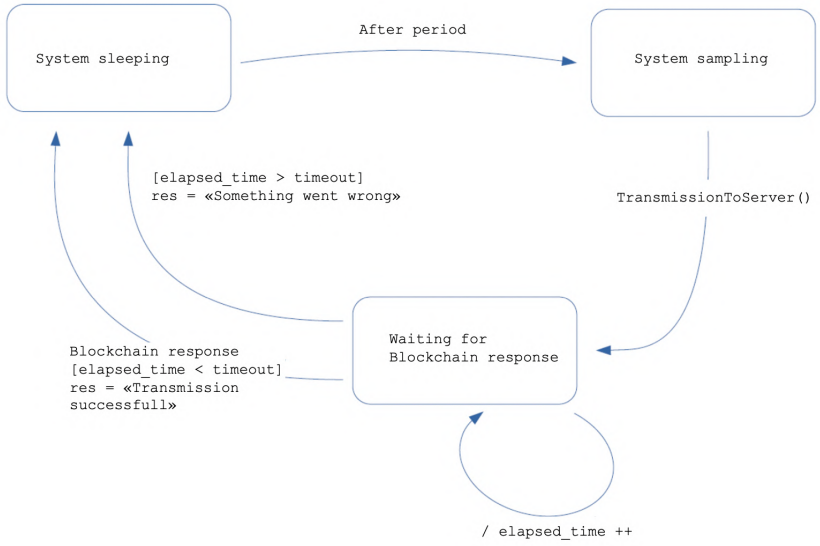


Figura 3.5: Macchina a stati per le fasi di campionamento e invio dati

viene utilizzato soltanto nel caso in cui i due dati presentino un'elevata differenza; in questo caso il sistema si accorge dello spike di temperatura non individuabile dal primo sensore e sostituisce la temperatura letta con quella del secondo, cercando di correggere l'errore.

3.2.3 Progettazione della componente software blockchain

Come giá accennato i nodi *IoT* non comunicano direttamente con la *blockchain* per questioni di scalabilitá e per evitare di dover effettuare pesanti interventi di manutenzione sull'intera schiera qualora sia necessario modificare la tecnologia *blockchain* utilizzata. Il Raspberry Pi 4 intermedio si occupa dell'interfacciamento con i *ledger* su cui vengono registrati i dati che sono, nel nostro caso, una *blockchain* privata realizzata mediante *Hyperledger Fabric* e la *DevNet IOTA*.

Il primo *ledger* é stato implementato utilizzando due computer fissi con processore dual-core Intel Celeron, dotati di 8GB di RAM. La scelta della CPU é stata indotta, oltre che da limitazioni tecniche di disponibilitá, anche dalla volontá di mostrare come, per gli scopi della *BCoT*, non sia necessario avvalersi di attrezzatura sofisticata con grande potenza di calcolo, ma sia sufficiente un dispositivo di uso comune. Ogni computer conteneva

un singolo nodo *blockchain*, con un singolo canale, due peer e cinque orderers.

Per quanto riguarda invece il collegamento con il *Tangle* della *DevNet* di IOTA questo é possibile direttamente tramite il Raspberry Pi 4. É da notare come questo tipo di collegamento sarebbe possibile anche direttamente sul Raspberry Pi Zero, senza la necessitá di interfacciarlo con il server intermedio; il motivo per cui non é stata perseguita questa strada é ancora una volta la ricerca di una maggiore scalabilitá, concentrando le eventuali modifiche future sull'implementazione della *blockchain* su un numero limitato di nodi piuttosto che su una schiera di questi. É infatti evidente come, nell'ottica di utilizzare N nodi di raccolta dati per ogni server intermedio (Figura 3.6), sia piú fruibile un sistema dove i dati vengano letti tramite stazioni che funzionano come "black-boxes" e sono del tutto trasparenti al tipo di *ledger* su cui essi vengono poi registrati.

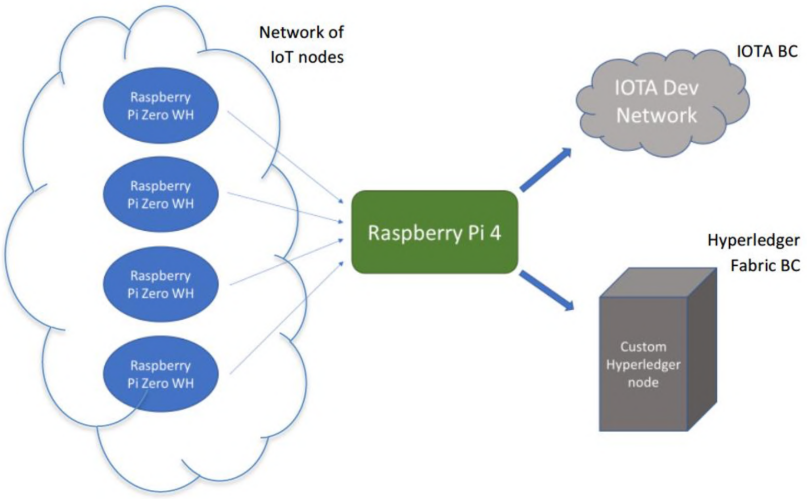


Figura 3.6: Topologia del sistema completo

La scelta di utilizzare entrambe le tecnologie é giá stata motivata precedentemente, ed é atta a creare un vero e proprio "trust-layer" sui dati, rendendoli visibili in maniera decentralizzata sulla *DevNet*. Per verificare qualsiasi dato sará quindi sufficiente recarsi su *Iota Tangle Explorer* [39] e ricercare l'*address* collegato ad uno dei nodi di raccolta dati. Scegliendo per esempio l'*address*:

```
MLNHJP9ODK9RDNBSMPYRRPEYZWFLCBUNIVEEFUDQILCP9DQCEEIKKTJ
SLEHRLPQVUODOZJOTGZPQ9LZUB
```

é possibile visualizzare una serie di transazioni, con relativo timestamp, risalenti ad un periodo tra Dicembre 2019 e Febbraio 2020, contenenti valori registrati da parte dei sensori. Queste transazioni sono registrate in maniera immutabile all'interno del *Tangle* della *DevNet*. I dati visualizzati all'interno del *Tangle* costituiscono una copia di quelli presenti all'interno dei nodi *Hyperledger*, e in totale quindi sono presenti cinque copie delle informazioni (una su *DevNet* e una su ognuno dei due peer presenti su ciascuno dei due computer). In Figura 3.7 viene riportato l'aspetto di una generica transazione ricercata attraverso il *Tangle Explorer*.

Figura 3.7: Visualizzazione del contenuto di una generica transazione sul Tangle Explorer

Come é possibile notale dalla Figura il dato viene scritto sul *Tangle* all'interno del campo *message*, che corrisponde al campo *signatureMessageFragment*, ovvero quello deputato a contenere i frammenti della firma del *bundle*. Il motivo per cui questo campo risulta disponibile per la scrittura é da ricercare nel fatto che sono state effettuate transazioni a valore nullo, e quindi il processo di firma di queste non é presente in quanto non vi é scambio di valore. Effettuando transazioni di questo tipo é possibile usare questo campo per inserirvi oggetti di tipo stringa o JSON, che vengono poi convertiti automaticamente in trytes per la fase di registrazione sul *Tangle* e poi di nuovo in stringa in fase di visualizzazione.

Per quanto riguarda invece l'implementazione dei nodi *Fabric*, é possibile accedere ai dati registrati su questi tramite il framework stesso di *Hyperledger* il quale, per maggiore chiarezza espositiva, permette di collegare il contenuto del nodo (e quindi della *blockchain*) ad un database *CouchDB* [40]. Questo consiste in un sistema gestionale di basi di dati non relazionale, facente uso di un linguaggio NoSQL per memorizzare dati sotto forma di oggetti JSON. La particolaritá del database é la facilitá di effettuarvi queries sotto forma di chiamate HTTP e la presenza di una interfaccia grafica intuitiva per la visualizzazione dei dati.

3.2.4 Progettazione dell'applicazione mobile

Per quanto riguarda la parte di visualizzazione dati e configurazione del sistema si é scelto di progettarla in ottica "mobile-first" e per questo motivo é stata realizzata un'applicazione mobile che permetta di interfacciare l'utente al sistema in maniera diretta e immediata. L'applicazione sviluppata é stata realizzata con un back-end in linguaggio *Javascript*, mentre il front-end é stato realizzato tramite *HTML/CSS* per poi effettuare il deploy specifico per ogni dispositivo.
La scelta della tecnologia é principalmente dovuta al fatto che si é cercato di rendere il progetto della parte mobile indipendente dal tipo di dispositivo su cui si vorrá effettuare l'effettivo deploy finale. L'applicazione é stata realizzata come fosse una webAPP e testata localmente su computer, per poi compilarla tramite Apache Cordova ed effettuare l'effettivo deployment su uno smartphone con sistema operativo Android. L'applicazione puó essere usata per tre differenti scopi, ognuno dei quali demandato ad un attore differente di una teorica supply-chain che utilizzi il sistema:

- Configurazione di un nodo;
- Inserimento di un blocco all'interno della *blockchain*;
- Monitoraggio dei dati presenti sulla *blockchain*.

Mentre le prime due operazioni sono pensate per un amministratore, che quindi viene reso in grado di aggiungere dinamicamente nuovi nodi di raccolta dati al sistema configurandoli al bisogno, oltre che di inviare una transazione manualmente alla *blockchain*, la seconda é pensata per un utente qualsiasi che voglia visualizzare i dati raccolti a scopo anche puramente di controllo.

Per configurare un nuovo nodo viene utilizzato un *seed* preimpostato per creare un'*address* sul *Tangle*, al quale verranno poi inviate le varie transazioni. Il sistema é strutturato come segue:

1. Viene generato un'*address* a partire dal seed e viene associato ad un nodo di raccolta dati specifico;

2. Viene registrata questa associazione su un altro *address* specifico sulla *DevNet*, la cui funzione é quella di "dictionary address", associando quindi ogni nodo all'indirizzo su cui verranno registrati poi i dati. L'*address* utilizzato é il seguente:

   ```
   VUJT9KRAAJQNPPNEIOLLPCQOHVUZMCSZBXFJCDGZIFODR
   NJAH9FSUKCFMQBHMK9JPBOCGTJASPQBLYGNXKGCQHOIXC
   ```

3. Quando il nodo viene acceso questo inizia a inviare pacchetti dati all'API server e questi vengono registrati sul *Tangle* all'indirizzo assegnato.

La parte *Javascript* del back-end viene utilizzata solamente per gestire le varie chiamate HTTP che le permettono di comunicare con l'API server.
Il funzionamento dell'applicazione si basa interamente sul plugin per lettura NFC disponibile per Apache Cordova, che viene utilizzato per leggere l'ID univoco di 7 byte presente su ogni tag, il quale corrisponde ad un valore esadecimale di 14 caratteri. Tramite questo valore, che viene comunicato dal nodo al server ogni volta che viene inviato un pacchetto dati, il server contatta il *dictionary* e recupera l'*address* a cui inviare la transazione, per poi registrarla sul *Tangle*. Il generico contenuto di una transazione registrata sul *dictionary* é infatti:

```
{
   "id": "427832FFFFFF9865FFFFFF81",
   "address": "MLNHJP9ODK9RDNBSMPYRRPEYZWFLCBUNIVEEFUDQILCP
   9DQCEEIKKTJSLEHRLPQVUODOZJOTGZPQ9LZUB"
}
```

Tramite questo, al server é reso noto l'*address* legato al nodo e automaticamente la transazione puó essere registrata (un esempio di transazione é riportato in Figura 3.7). La stessa logica viene utilizzata anche nella fase di lettura dei dati, richiedendo all'API server di fornire al dispositivo mobile

tutte le transazioni dell'*address* desiderato.

Per un corretto funzionamento del sistema all'apertura dell'applicazione il dispositivo richiama l'evento *ondeviceready*, tipico di Cordova, attivando la lettura del sensore NFC al fine di notificare all'utente qualora questo fosse non attivo o non disponibile. Una volta che la lettura é stata abilitata l'utente puó avvicinare il dispositivo al tag NFC allegato alla scheda di raccolta dati e richiedere di effettuare una qualsiasi operazione tra le tre specificate sopra. Indipendentemente dall'operazione il passo successivo é la chiamata della route corrispondente sull'API server, tramite *XMLHttpRequest*, inviando un pacchetto in formato JSON che specifichi i dati da trasferire. In Figura 3.8 é riportata a titolo di esempio la schermata utilizzabile per configurare un nuovo nodo di raccolta dati.

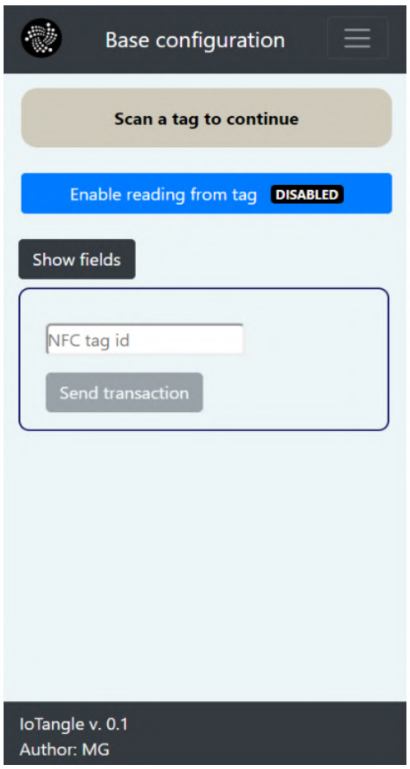


Figura 3.8: Schermata di configurazione di un nuovo nodo

3.2.5 Progettazione del server

Per quanto riguarda l'API server questo ha una duplice funzione. In prima istanza permette di collegare le stazioni di raccolta dati alla *blockchain Fabric*, effettuando un semplice reindirizzamento dei dati all'indirizzo corrispondente ai computer che ospitano i *peer*; secondariamente viene utilizzato per gestire l'interfacciamento dell'applicazione mobile e dei nodi con la *DevNet*. Questa seconda operazione presenta criticitá maggiori poiché, mentre per *Fabric* le operazioni da eseguire e lo *smart-contract* sono gestiti a livello del nodo finale (che per i fini di questo elaborato sará trattato come una "black-box"), per *IOTA* é il server stesso che si occupa di gestire le transazioni e le richieste.

Il server é stato realizzato mediante la libreria *express*, che permette di utilizzare *Javascript* per esporre API raggiungibili tramite chiamate HTTP. Inoltre l'utilizzo di *Javascript* permette di implementare direttamente le funzioni della libreria *iota.lib.js* per il collegamento al *Tangle*, la quale richiede di ragionare in ottica server piuttosto che client. Lo script é in perenne esecuzione sul Raspberry Pi 4 e, poiché gestire contemporaneamente i flussi di dati provenienti dai nodi e quelli di richieste provenienti dall'applicazione mobile comporterebbe latenze inaccettabili, sono stati suddivisi due script in esecuzione contemporaneamente e contattabili su due porte differenti; in questo modo i nodi possono mandare continuamente dati e le richieste che provengono dall'applicazione sono gestite in maniera differente ma parallela.

Una delle maggiori complicazioni emerse all'interfacciamento con *IOTA*, nella parte di richiesta dei dati, é stato il fatto che la comunicazione con il *Tangle* deve essere necessariamente gestita in maniera asincrona, in quanto non é possibile prevedere con accuratezza i tempi di risposta alle varie richieste. Utilizzando nodi pubblici, quindi contattati contemporaneamente anche da altri utenti, la latenza assume un grado di aleatorietáelevato e possono teoricamente passare diversi secondi tra una richiesta e la risposta. Il sistema quindi viene lasciato in stand-by alcuni secondi ogni volta che viene richiesta una serie di dati, posticipando l'invio della *response* finché non si é in grado di allegarvi dati. Per rendere questo possibile l'invio della *response* é stato spostato all'interno della *callback* delle varie funzioni di libreria, la quale viene richiamata solo una volta che il metodo asincrono ha prodotto un risultato.
Questo tipo di operazioni non é stato necessario invece per la parte di invio dei dati alla *DevNet*, anche se troviamo comunque un tempo aleatorio di attesa, che verrá analizzato succesivamente.

3.3 Test delle performance del sistema

Il testbed di laboratorio é stato mantenuto in funzionamento per diverse settimane, utilizzando appunto i due computer descritti precedentemente e undici nodi IoT tutti con le stesse caratteristiche presentate. Il numero di dati registrato totale é stato di circa mezzo milione, per un'occupazione di spazio su disco totale, per ogni computer, pari a $15.45GB$, che deve essere effettivamente divisa per due per calcolare l'effettiva occupazione della singola copia della *blockchain* costruita con *Fabric*, poiché ogni peer ne mantiene una copia completa; in tutto quindi il singolo *ledger* aveva un peso di $7.72GB$, decisamente contenuto rispetto al volume di dati.

Alcune considerazioni vanno fatte sui valori di occupazione di spazio gestibili per il sistema. Considerando un periodo di campionamento dei dati dai sensori pari ad un minuto abbiamo, ogni giorno, $60 \cdot 24 = 1440$ transazioni; per raggiungere mezzo milione impiegheremmo quindi circa un anno, occupando soltanto una percentuale pari a 1.54% (contando di mantenere due peer su ogni computer) dello spazio di archiviazione totale di un comune hard-disk da $1TB$.
Per casi reali di raccolta dati, al netto di situazioni eccezionali dove sia necessario raccogliere centinaia di dati al secondo per effettuare controlli estremamente precisi, possiamo pensare ad un periodo di campionamento non di molto distante da quello impostato in fase di test; qualora decidessimo di campionare dati una volta al secondo (valore in contrasto con le specifiche del sensore di temperatura, che richiede di campionare al massimo con periodo di circa due secondi, per le motivazioni giá illustrate) avremmo un numero di transazioni al giorno pari a 86400, quindi mezzo milione in circa cinque giorni. In questo caso lo spazio totale occupato in un anno sarebbe di poco superiore a $560GB$ per ogni copia, occupando quindi un comune hard-disk da $1TB$ all'anno.
Da questi dati preliminari é quindi evidente come, almeno per i ratei di transazione definiti per la fase di test, la possibiliá di esaurire lo spazio di archiviazione non rappresenti un collo di bottiglia.

Per quanto riguarda invece il TPS del sistema é necessario effettuare considerazioni diversificate per *IOTA* e per *Hyperledger*, e proprio per questo motivo le due implementazioni sono state analizzate in maniera separata, e unite soltanto in ultima istanza.

3.3.1 Test dei nodi Hyperledger Fabric

Partendo da *Hyperledger*, il dato migliore ottenuto, considerando di spegnere la parte di invio alla *DevNet* e concentrarsi solo sull'ottenere il miglior TPS possibile per nodi *Fabric*, é stato di trentacinque transazioni al secondo gestibili per ogni nodo (nello specifico ogni secondo al nodo venivano inviati cinque blocchi contenenti ognuno sette transazioni). Per rendere possibile questo test ovviamente non sono stati utilizzati i nodi *IoT* di raccolta dati ma si é deciso di generare direttamente a livello di API server uno stream di transazioni, al fine di verificare i limiti del sistema.
Per valori superiori il sistema presenta un comportamento particolare: per qualche secondo dopo l'accensione il nodo é in grado di gestire l'altissima mole di transazioni, per poi riportare una serie di errori nella fase di registrazione dei blocchi sul *ledger*. Questo puó essere, almeno a livello teorico, imputato all'hardware utilizzato, poiché una CPU dual-core non é in grado di fornire l'alto livello di parallelizzazione dei processi necessario a sostenere un TPS cosí alto. Questa caratteristica é richiesta poiché l'algoritmo di consenso di *Fabric* richiede che il *peer* simuli ogni transazione prima di passarla ad un *orderer*, quindi il tempo necessario per registrare una transazione risulta pari al doppio di quello che sarebbe effettivamente necessario al sistema per registrarla singolarmente, limitando il TPS massimo raggiungibile. A livello pratico, oltre il TPS indicato, il lavoro della CPU si attesta a valori vicini al 100%, causando il fenomeno del "thermal throttling", il quale riduce la frequenza di lavoro della CPU per permetterle di raffreddarsi. Questo tuttavia rende il sistema non piú in grado di gestire il numero di transazioni ricevute e compaiono gli errori di cui prima, i quali portano al degrado delle prestazioni, rendendo impossibile anche la registrazione di una singola transazione. Poiché per recuperare il corretto funzionamento il sistema deve essere spento e *orderer* e *peers* devono essere riavviati, questa situazione é da evitare assolutamente per garantire un funzionamento costante del sistema.

Per migliorare le prestazioni del sistema é possibile utilizzare un computer equipaggiato con una CPU che presenti piú cores e che quindi possa garantire un maggiore livello di parallelizzazione. Alcuni test effettuati con un computer dotato di CPU Intel quad-core hanno mostrato come sia possibile raggiungere TPS pari a 100. Il valore del TPS sembra, quindi, linearmente dipendente dalla capacitá di parallelizzazione del sistema e quindi dal numero di core disponibile per la computazione dei blocchi e la loro registrazione.

Un altro parametro investigato é stata la RAM del computer, poiché i test preliminari hanno mostrato un'occupazione di questa in maniera periodica,

con picchi fino a valori vicini al 100% e periodi di scarsa occupazione. Questi risultati sono da imputare al fatto che, durante il periodo di simulazione della transazione, i *peer* sfruttano la RAM per conservare le informazioni intermedie e le transazioni che aspettano approvazione, per poi liberarla al momento in cui queste vengono registrate. Sebbene da questo punto di vista gli $8GB$ del computer equipaggiato con Interl Pentium Dual-Core si siano rivelati sufficienti per non incorrere in problematiche a livello prestazionale é da sottolineare come, per evitare di raggiungere picchi vicini al 100% che potrebbero degradare il sistema su tempi medio-lunghi, siano consigliati almeno $16GB$ di RAM se si vogliono mantenere tali TPS.

Ultima considerazione da fare riguarda un'altra implementazione di *Hyperledger*, quale *Sawtooth*, che si é dimostrata, in alcuni test pubblicati, capace di raggiungere TPS maggiori. Si é comunque scelto di mantenere *Fabric* per il suo alto livello di personalizzazione e per il fatto che, nei casi studio analizzati, non é mai emersa le necessitá di raggiungere valori di TPS pari o superiori a qualche centinaio. Contrariamente, la personalizzazione risulta uno dei requisiti fondamentali emersi in fase di analisi.

3.3.2 Test di IOTA

Per il test della parte relativa all'interfacciamento con la *DevNet* si é scelto di utilizzare un singolo nodo, poiché il collo di bottiglia della comunicazione con il *Tangle* si trova a livello di interfacciamento del server con quest'ultimo e non al livello di connessione nodo-server (sempre supponendo di poter gestire ogni transazione in parallelo). Utilizzare quindi un singolo nodo o tutti e undici i nodi contemporaneamente non avrebbe comportato alcuna alterazione nei dati e quindi si é scelto di procedere con la scelta che fornisse una visualizzazione piú chiara. Per stressare il sistema é stato eliminato il delay presente tra le transazioni, per cui ognuna veniva inviata dal nodo di raccolta dati al server esattamente una volta ricevuto l'ACK della precedente da parte del secondo. I parametri analizzati sono:

- Tempo necessario per la misura con tutti i sensori, T_{base}^{MIS};
- Tempo necessario al server per ricevere il pacchetto, contattare il *Tangle*, registrare la transazione e restituire risposta positiva al nodo, T_{iota}^{VALID}.

I due valori sono stati misurati varie volte su un arco temporale pari a cinque minuti e i risultati sono visualizzati in Figura 3.9, e i valori in Figura 3.11. Come é possibile vedere dai risultati del test il tempo necessario per completare un ciclo di lettura e scrittura dati é fortemente dipendente dal

tempo necessario al nodo pubblico per confermare la ricezione corretta e la trasmissione del pacchetto al *Tangle* della *DevNet*.

Un ulteriore test é stato eseguito supponendo di aggiungere una nuova stazione di raccolta dati ma di non dotare il server di strumenti atti a gestirla parallelamente alla prima: in questa situazione, almeno teoricamente, i tempi dovrebbero raddoppiare in quanto il server deve attendere la conferma della prima di inviare la seconda. I risultati, compatibili con le considerazioni teoriche, sono riportati in Figura 3.10, e i valori in Figura 3.12.

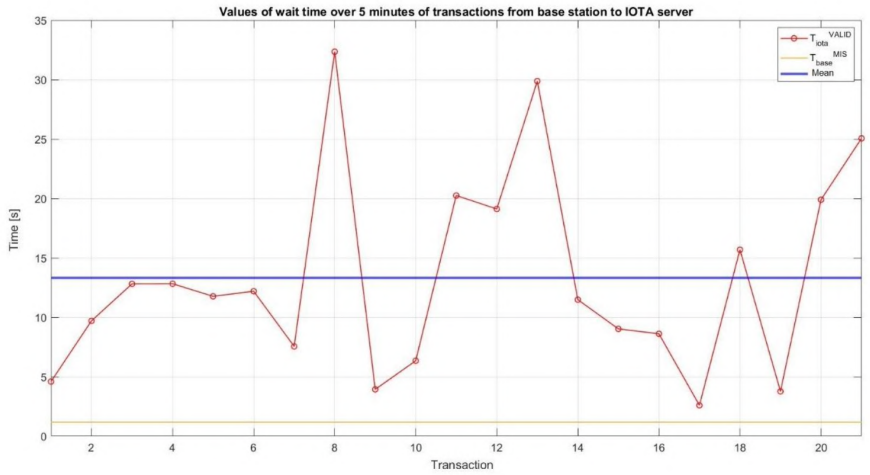


Figura 3.9: Risultati del test con singolo nodo collegato

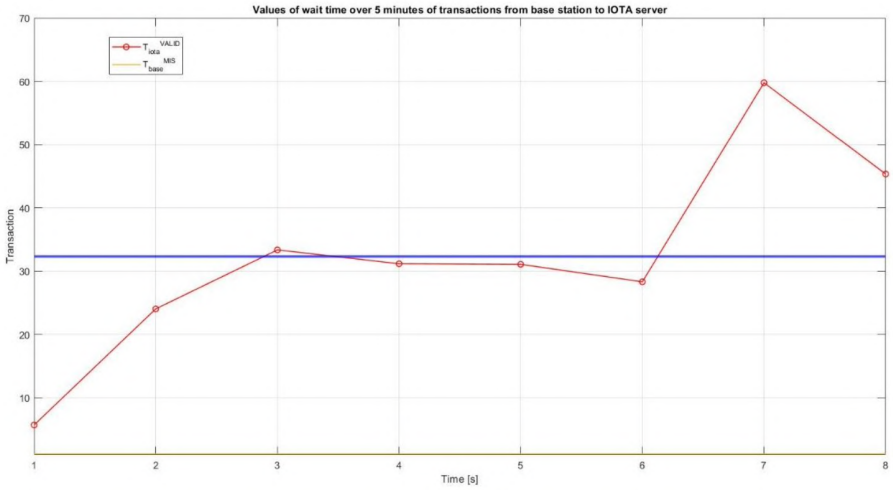


Figura 3.10: Risultati del test con due nodi collegati

	T_{iota}^{VALID}			T_{base}^{MIS}
Singolo nodo	Max	Min	Medio	Costante
embedded	32.35s	2.6s	13.31s	1.81s

Figura 3.11: Tabella riferita alla configurazione con singolo nodo

	T_{iota}^{VALID}			T_{base}^{MIS}
Due nodi	Max	Min	Medio	Costante
embedded	59.77s	5.74s	32.6s	1.81s

Figura 3.12: Tabella riferita alla configurazione con due nodi

Come ci si aspettava i valori risultano circa raddoppiati, e questo crea particolare criticitá sopratutto per quanto riguarda la scalabilitá del sistema, poiché non é pensabile che l'inserimento di un nuovo nodo di raccolta dati raddoppi la latenza del sistema. Per risolvere questo problema é possibile in prima istanza impostare un *delay* alle transazioni pari almeno a qualche minuto, assicurandosi cosí che due transazioni non siano mai gestite contemporaneamente. Qualora questa soluzione non fosse possibile puó essere adottata la parallelizzazione delle richieste su piú canali all'interno del server, in modo da poter gestire piú richieste contemporaneamente in modo asincrono.

Per quanto riguarda invece la fase di recupero dei dati é stato calcolato che mediamente un pacchetto in formato JSON ha il peso di $200byte$. Al sistema é stato richiesto di scaricare un numero fisso di pacchetti, pari a 150, dal *Tangle*, il cui peso totale ammonta quindi a circa $30kByte$, valutando il tempo impiegato. Come é evidente dalla Figura 3.13 il tempo risulta abbastanza variabile, con picchi di quasi un minuto, che tuttavia sono stati registrati solo occasionalmente. Il valore medio riscontrato, effettuando piú test su un intervallo fisso di cinque minuti, é pari a $5.1s$ ed é quindi mediamente un buon valore, se si considera che questi dati necessitano di essere visualizzati da parte dell'utilizzatore di un'applicazione mobile.

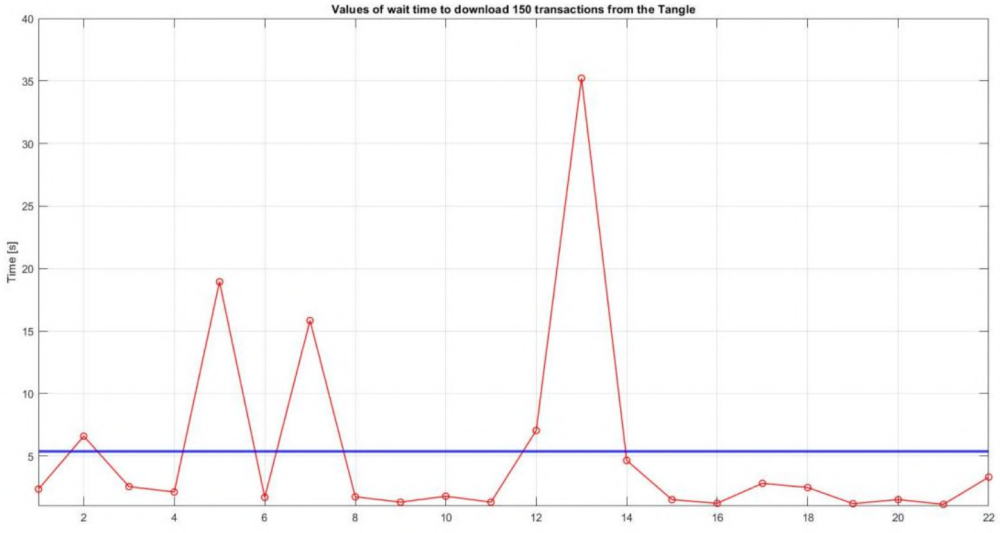


Figura 3.13: Tempo necessario per scaricare 150 transazioni

3.3.3 Test con nodo Hornet privato

I dati forniti finora rappresentano il caso in cui si decida di delegare ad un nodo esterno il compito di inviare i dati al *Tangle*, e quindi non si disponga di un nodo *Hornet* personale. Questa seconda ipotesi tuttavia permetterebbe, almeno in linea teorica, di migliorare la prestazione del sistema, non dovendo piú concorrere con altri utenti per l'utilizzo del nodo condiviso. Per tale motivo, in fase finale di test, si é deciso di implementare un nodo *Hornet* su un computer con sistema operativo Linux, collegandovi anche un'istanza di *Chronicle* per avere un vero e proprio database che conservasse le transazioni al di fuori del *Tangle*.

La configurazione del nodo ha reso necessario effettuare alcune scelte, che sono state imposte dall'attuale implementazione dei nodi *Hornet* la quale, almeno per fini industriali, non é ad oggi in grado di garantire alcune caratteristiche fondamentali per un deployment su larga scala.
Nello specifico non viene ancora reso possibile utilizzare il *Chronicle* in maniera selettiva ma questo é configurabile solamente per salvare in locale tutte le transazioni che vengono fatte sul *Tangle* a cui é collegato il proprio nodo *Hornet*; la conseguenza é facile da comprendere: se vogliamo collegare il nodo al *Tangle* pubblico avremo, oltre che le nostre transazioni, anche tutte le altre transazioni inviate da altri utenti salvate in copia locale. In alternativa possiamo decidere di configurare una rete privata, con un *Tangle* che sia collegato solo a nodi di nostra proprietá o nodi che vengono configurati e accettati nel network; in questo modo il *Chronicle* manterrá tutte le transazioni presenti su questo *Tangle* privato ridotto, contenendo l'occupazione di spazio di archiviazione alle sole transazioni di interesse per il progetto.

Entrambe le configurazioni presentano delle problematiche non trascurabili:

- La configurazione con nodo *Hornet* collegato al *Tangle* pubblico, con istanza *Chronicle* non selettiva, non permette di selezionare e salvare in maniera perenne soltanto le transazioni proprie, ma costringe a salvare copia integrale del *Tangle*, con conseguenti problemi di occupazione di memoria;

- La configurazione con nodo *Hornet* privato permette di risolvere la problematica dello spazio di archiviazione ma é in contrasto con il requisito numero uno per cui é stata scelta la configurazione ibrida *Hyperledger-IOTA*, cioé la necessitá di appoggiarsi ad una infrastruttura pubblica oltre che ad un privata.

La necessitá di utilizzare un nodo *Chronicle* é principalmente dettata dal fatto che, come giá accennato in precedenza, la rete *IOTA* effettua regolarmente degli *snapshot* in cui viene completamente svuotato il *Tangle*. Collegandosi quindi tramite l'API fornita dalle varie librerie ai nodi pubblici si ha sempre una visione parziale del *Tangle*, limitata allo *snapshot* corrente. La stessa cosa avviene con un nodo *Hornet* personale collegato al *Tangle* pubblico, poiché questo, qualora non fosse presente un'istanza *Chronicle*, manterrebbe sempre e solo la copia del *Tangle* e quindi seguirebbe la logica degli *snapshot*. Aggiungere un *Chronicle* al nodo ci permette, ogni volta che viene effettuato uno *snapshot*, di mantenere comunque una immagine completa del *Tangle* sul nodo personale, anche se l'istanza condivisa sulla rete risulta diversa.

Il problema non si presenta collegandosi al *Tangle* tramite *Iota Tangle Explorer* [39], poiché il nodo a cui questo sito fa riferimento per le varie queries é dotato di un *Chronicle* in esecuzione da vari *snapshot*.

Il problema dell'utilizzo di un nodo *Hornet* personale senza uso di *Chronicle* diventa ancora piú critico qualora decidessimo di usare la *DevNet* piuttosto che la *MainNet*, come in realtá é interesse per i fini di questo progetto. In questo caso gli *snapshot* hanno cadenza circa giornaliera e quindi transazioni vecchie di soli due giorni risulterebbero giá non disponibili tramite la libreria, il che é del tutto incompatibile con le nostre necessitá. Una soluzione potrebbe essere quella di non utilizzare la libreria ma di collegarsi direttamente al nodo *Hornet* usato per mantenere il *Tangle Explorer*, ma questa configurazione ha necessitá di essere studiata approfonditamente e rappresenta comunque un "work-around" non implementabile in ottica scalabile di produzione.

Ad oggi l'unica soluzione accettabile sembra essere un'implementazione di *Chronicle* selettivo, che sia in grado di collegarsi al *Tangle* pubblico ma salvi in locale in maniera perenne solo le transazioni provenienti dal nodo *Hornet* a cui si trova collegato. Dal confronto diretto con la *IOTA Foundation* é emerso come questa ipotesi risulti nella lista dei prossimi aggiornamenti.

Per quanto riguarda i test effettuati si é quindi scelto di optare per una soluzione con singolo nodo *Hornet*, collegato alla rete pubblica, corredato da un'istanza di *Chronicle* non selettivo. I test sono stati effettuati sia collegandosi alla *MainNet* che alla *DevNet* e i risultati sono decisamente confortanti:

- Per quanto riguarda la *DevNet* i tempi necessari a processare le transazioni scendono sotto il secondo, e il massimo TPS raggiungo é di circa 10 transazioni al secondo, abbastanza stabile in diversi test effettuati;
- Per quanto riguarda la *MainNet* i tempi salgono leggermente e non abbiamo piú lo stesso livello di stabilitá che troviamo nel collegamento con la *DevNet*. Il TPS medio sale a meno di una transazione al secondo, e la latenza mostra una certa variabilitá nei vari test.

I risultati di cui sopra sono da investigare ulteriormente ma questo non é stato ad oggi possibile a causa dell'attuale versione del nodo *Hornet*, la quale é ancora particolarmente instabile e, almeno sull'hardware a disposizione, tende a causare malfunzionamenti e disallineamento della copia locale del *Tangle* rispetto a quella pubblica, il ché si riflette nell'impossibilitá di inserirvi transazioni con continuitá e nella necessitá di riavviare periodicamente il sistema.

3.4 Alcune considerazioni finali

Dai dati emersi in fase di testing é evidente come questo non sia ancora pronto per un deployment effettivo su larga scala, almeno nei termini definiti finora. Le principali problematiche che sono emerse riguardano:

1. Insostenibilitá di *IOTA* per TPS alti: non é possibile utilizzare nodi pubblici per gestire alti ratei di transazioni e i nodi *Hornet* personali hanno ancora problemi di stabilitá per poter garantire un funzionamento duraturo e costante;
2. Complessitá di gestione di *Fabric*: le specifiche di *Fabric* lo rendono adatto a realizzare *blockchain* di elevata complessitá e il sistema risulta inutilmente complesso per un uso cosí semplificato, in mancanza di canali e organizzazioni. Inoltre *Fabric* non offre un modo intuitivo per gestire *digital-twins* e passaggi di proprietá di informazioni e asset;
3. Consumo di corrente troppo elevato da parte delle stazioni di raccolta dati: i nodi infatti sono, in questa configurazione, alimentati a corrente continua. Questo é necessario poiché i Raspberry Pi, a causa sopratutto della presenza del modulo WiFi e del SO, non sono alimentabili a batteria per piú di qualche giorno di fila. Non é quindi pensabile un'alimentazione solare per dispositivi con questo consumo energetico, o per lo meno questa soluzione non risulta affidabile sul lungo periodo.

Il primo problema risulta di risoluzione relativamente semplice: per applicazioni come quelle analizzate il TPS tipico richiesto é pari a qualche transazione all'ora, quindi é sufficiente aumentare il ritardo di invio di queste. Per quanto riguarda invece la terza criticitá si ritiene sufficiente sostituire le stazioni di raccolta dati basate su Raspberry Pi Zero con dispositivi senza SO e che comunichino con il server tramite interfaccia RF piuttosto che WiFi (questa soluzione sará presentata in breve nel capitolo successivo). Il problema di cui al punto due é invece piú complesso e richiede di ripensare il sistema sostituendo la *blockchain* realizzata mediante *Hyperledger* con una sua implementazione alternativa; anche questo verrá discusso piú avanti.

In un ipotetico scenario di deployment su larga scala é ammissibile pensare ad un sistema di monitoraggio con decine di stazioni di raccolte dati distribuite su vasta area, tutte comunicanti con un unico server, oppure un sistema in cui ogni oggetto fisico sia dotato di una componente hardware interna atta a raccogliere dati sul suo utilizzo e inviarli autonomamente alla *blockchain*. In questo scenario i sistemi proposti finora risultano ancora da modificare, prevedendo la possibilitá che i nodi stessi comunichino direttamente con la *blockchain*, senza bisogno di un server centrale, oltre che l'uso di componenti hardware a basso consumo.

3.5 Caso di studio: implementazione per tracciamento di frigoriferi industriale

Durante lo sviluppo del progetto *BlockWEEE* é stato di grande importanza il caso studio del tracciamento di un frigoriferi professionali. Infatti, per una grande multinazionale operante su scala mondiale, il problema dello smarrimento e dello scorretto smaltimento di apparecchiature industriali date in dotazione a bar, ristoranti, centri ricreativi e simili rappresenta ad oggi una criticitá all'interno dell'ecosistema industriale: lo smaltimento non corretto dei frigoriferi rientra nella responsabilitá estesa del produttore e potrebbe rappresentare per l'azienda un punto debole per l'immagine di sostenibilitá che essa vuole comunicare.

In questa sezione verrá presentata una proposta di adattamento del sistema fino a quí descritto per monitorare il processo di vita di un frigorifero industriale dal momento del posizionamento all'interno dell'attivitá. Il progetto é presentato in qualitá di Proof of Concept (PoC), quindi tutte le considerazioni effettuate riguardano solo la fattibilitá dello stesso e non é

presente un prototipo funzionante specifico. É inoltre da sottolineare come non si consideri la fase di costruzione e distribuzione del frigorifero ma lo scopo della proposta sia solamente il monitoraggio dello stato dello stesso all'interno dell'attivitá, al fine di intercettare qualora questo venga spostato, venduto o ceduto in maniera scorretta.

3.5.1 Spiegazione del caso studio

I frigoriferi ad uso industriale sono distribuiti dell'azienda ai proprio esercizi partner in modalitá di comodato d'uso: le attivitá li ricevono gratuitamente e sono obbligate dal contratto ad acquistare prodotti direttamente dall'azienda da alloggiarvi all'interno, per rivenderli ai propri clienti e fare indirettamente pubblicitá all'azienda. Inoltre, sempre a livello di contratto, é specificato come l'esercizio sia obbligato a mantenere all'interno del frigorifero almeno una percentuale fissata di prodotti a marchio dell'azienda, sempre per motivazioni di pubblicitá e visibilitá del marchio.
Per il rifornimento di questo tipo di apparecchiature é previsto l'intervento periodico di un commerciale, il quale si reca presso l'esercizio per raccogliere gli ordini e far consegnare la merce in base al consumo nel determinato periodo di tempo; questa visita fisica serve anche a controllare che il frigorifero sia ancora utilizzato e presente nell'esercizio e che i prodotti del marchio vi si trovino nella corretta percentuale. L'intervento puó avere cadenza piú o meno frequente durante l'anno, dipendente strettamente dalle prestazioni dell'esercizio in termini di prodotti dell'azienda che vengono venduti. Questa modalitá risulta ad oggi particolarmente scomoda in quanto poco digitalizzata; richiede infatti un continuo intervento umano nel recarsi presso le attivitá per controllare che tutte le norme contrattuali siano rispettate. Oltre a questo presenta anche problematiche logistiche nel momento in cui l'attivitá faccia poco uso dei prodotti e quindi la periodicitá della visita sia superiore a qualche mese, o addirittura annuale. In questi ultimi casi é frequente il caso in cui, al momento della visita da parte del commerciale, l'attivitá abbia per esempio chiuso, o cambiato gestione, e quindi il frigorifero non sia piú in sede e diventi difficoltoso rintracciarlo. Questo apre peró il problema della gestione del suo fine vita poiché, qualora non venisse smaltito correttamente, la responsabilitá rimarrebbe comunque dell'azienda e non dell'attivitá che lo possedeva in comodato d'uso.

Dall'azienda é quindi stato richiesto un sistema che potesse aiutare nell'operazione di monitoraggio, garantendo un beneficio agli esercenti che si comportassero in maniera corretta e permettendo di monitorare lo stato dei frigoriferi senza l'effettivo intervento periodico.

3.5.2 Esposizione della soluzione proposta

La soluzione proposta prevede l'integrazione di una stazione di raccolta dati ambientali, posizionata all'interno del frigorifero, con dati provenienti dai telefoni cellulari degli utenti dell'attivitá. Piú nello specifico il monitoraggio avviene in due fasi:

1. Il frigorifero é dotato dalla nascita di un sistema di controllo della temperatura, di un localizzatore GPS e di un sistema inerziale a 6 assi, in grado di comprenderne la posizione spaziale;

2. Sul frigorifero é apposto un QR code univoco, o tag NFC, che gli utenti possono scansionare per registrare dati riguardanti la specifica apparecchiatura.

Grazie alla commistione di questi due sistemi é possibile intercettare gli spostamenti del frigorifero in primis, mentre si é anche in grado, tramite i feedback degli utenti, di monitorare l'uso del frigorifero da parte dell'attivitá e quindi di capire quanti prodotti del marchio vengono venduti abitualmente.

Punto di studio interessante é la modalitá di alimentazione della scheda di raccolta dati presente all'interno del frigorifero. Poiché si suppone che questa debba raccogliere costantemente dati e registrarli su un *ledger*, che verrá chiarito in seguito, é necessario dotare il sistema di una fonte di alimentazione continua. Il problema é relativamente trascurabile nel momento in cui il frigorifero si trovi in posizione all'interno dell'attivitá, poiché si suppone che sia alimentato dalla rete elettrica del locale e quindi la scheda di raccolta dati sia alimentabile di conseguenza. Per essere efficiente anche qualora il frigorifero venisse spostato, il sistema puó essere progettato a priori per reagire ad una serie di stimoli (per esempio il superamento di una certa soglia di accelerazione) per comunicare la sua posizione e informare sul proprio stato. In questo modo sarebbe per esempio possibile intercettare uno spostamento tramite un camion, rendendo possibile per l'azienda accertarsi del perché il frigorifero sia stato spostato. Un altro sistema di allarme potrebbe essere inserito per intercettare la mancanza dell'alimentazione per un periodo di tempo determinato (o per esempio intercettando un aumento della temperatura fino a quella ambiente), il quale starebbe ad indicare che l'attivitá potrebbe aver chiuso o il frigorifero potrebbe essere stato dismesso; questo sarebbe possibile prevedendo un meccanismo di alimentazione di backup a batteria della durata di qualche giorno.

In tutte le altre situazioni la stazione di raccolta dati si troverebbe in stato di risparmio energetico, con interrupt collegati alle varie periferiche, configurati per intercettare un comportamento come quelli di cui sopra. I dati raccolti dalla scheda dovrebbero poi essere trasferiti all'interno di un *ledger* distribuito in grado di registrarli in maniera immutabile e permanente, possibilmente collegandoli all'attivitá all'interno della quale il frigorifero si trova. Questo processo, che puó tranquillamente essere effettuato usando il *Tangle* di *IOTA*, sarebbe abilitato dall'inserimento all'interno della stazione di raccolta dati di un modulo GMS, comunicante via rete cellulare con un server centrale che recuperi le informazioni e le registri sul *ledger*.

Accanto a questo sistema automatico é stato previsto l'intervento manuale dei clienti dell'attivitá, i quali, tramite un'applicazione mobile specifica, o sfruttando una funzionalitá da integrare in un'APP giá distribuita dall'azienda, possono scannerizzare il QR code e fare una foto al frigorifero, da immettere poi nella *blockchain*. Queste foto, analizzate da un opportuno software di riconoscimento immagini, possono essere usate per confermare la presenza del frigorifero all'interno del locale, combinandole con i dati sulla posizione GPS forniti dal telefono al momento dell'invio. É inoltre possibile, tramite le foto, valutare lo stato di utilizzo dell'apparecchio e la conformitá con le norme contrattuali.

Il movente per l'utilizzo dell'intero sistema potrebbe essere l'inserimento di una serie di token, ottenibili dagli utenti per l'uso di questa funzionalitá, la cui raccolta potrebbe portare alla vincita di gadget, o che potrebbero essere scambiati con prodotti omaggio. Contemporaneamente il proprietario dell'attivitá vedrebbe il proprio operato ricompensato in base alla quantitá di persone che partecipa al progetto, ricevendo anch'egli gadget per il proprio locale o forniture scontate o gratuite, al raggiungimento di determinate soglie. Il tutto sarebbe sempre garantito e controllato da uno smart contract, in grado di valutare la conformitá delle fotografie e assegnare i token, nonché di registrare i dati provenienti dagli utenti sulla *blockchain*. In Figura 3.14 é mostrato un esempio grafico del funzionamento logico del sistema.

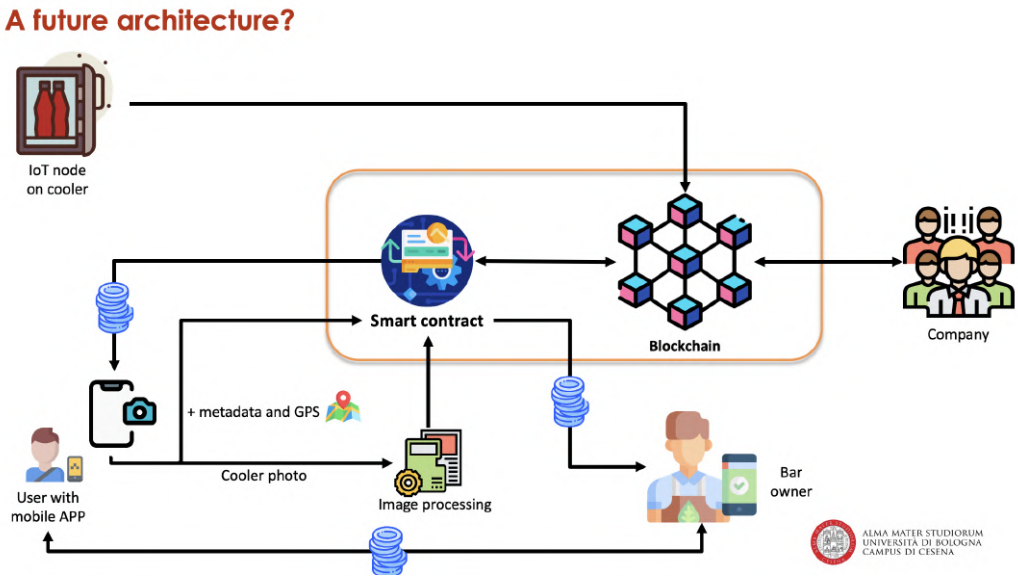


Figura 3.14: Schema logico del sistema di monitoraggio di un frigorifero

Capitolo 4

Oltre BlockWEEE: due proposte per il superamento delle criticitá emerse

I risultati ottenuti a seguito dell'implementazione del test-bed del progetto *BlockWEEE* hanno aperto la strada alla possibilitá di applicare alcune modifiche al sistema al fine di superare le criticitá emerse. In prima istanza infatti si é mostrato come realizzare il sistema tramite Raspberry Pi possa non essere la soluzione migliore in termini di efficienza energetica, oltre che di costo. In secondo luogo si é valutato come *Hyperledger Fabric* potrebbe essere sostituito da una implementazione di *blockchain permissioned* piú semplice, oltre che piú adatta a gestire asset di tipo "digital-twin".

In questo capitolo finale sará proposta una soluzione per ognuno dei due problemi: verrá quindi mostrata una versione alternativa della stazione di raccolta dati, piú economica e senza necessitá di comunicazione WiFi; parallelamente, per quanto riguarda l'implementazione della *blockchain*, verrá proposta una tecnologia alternativa ad *Hyperledger*.

Le soluzione qui proposte sono ancora in fase di studio, e costituiscono ad oggi piú un PoC che un'effettiva implementazione. Per questo motivo non verranno presentati test approfonditi e le considerazioni fatte sul miglioramento delle prestazioni del sistema saranno di carattere quasi puramente teorico. I lavori futuri che prenderanno spunto da questo elaborato potranno avvalersi delle considerazioni effettuate fino a questo punto, andando ad analizzare nel dettaglio la parte di alimentazione e sostenibilitá a lungo termine che é stata tralasciata in questa seconda parte. L'obiettivo di realizzare un vero e proprio "trust-layer" appoggiato su una infrastruttura $BCoT$ é ancora

chiaro ed é stato dimostrato empiricamente come questo sia fattibile, almeno su piccola scala.

4.1 Sostituzione dei Raspberry Pi Zero e Raspberry Pi 4

Come giá mostrato, la principale problematica dell'utilizzo dei dispositivi della famiglia Raspberry sta nella difficoltá di alimentarli a batteria o in generale tramite sistemi ricaricabili (per esempio usando un pannello fotovoltaico). Questa caratteristica ha posto l'inevitabile problema di dover garantire una fonte di alimentazione continua alle schede in fase di sviluppo e test. Sebbene all'interno dei laboratori dell'Universitá di Bologna l'alimentazione non abbia costituito un effettivo problema, in un ottica di deployment su larga scala ed uso in ambiente di produzione é impossibile garantire che i dispositivi possano essere collegati costantemente all'alimentazione, ed é quindi necessario prevedere la possibilitá di alimentarli a batteria o tramite fonte alternativa. Inoltre, solitamente i dispositivi IoT che effettuano monitoraggio sono pensati per funzione in modalitá di risparmio energetico, rimanendo spenti la maggior parte del tempo e riaccendendosi periodicamente al momento di campionare i dati; questo é incompatibile con la presenza del SO, che costringe il dispositivo ad effettuare una serie di operazioni di accensione prima di essere effettivamente operativo.

4.1.1 Da Pi Zero ad Arduino Nano

Il primo passo fatto per realizzare schede di raccolta dati con consumo energetico e costi contenuti, garantendo inoltre la possibilitá di funzionare in risparmio energetico, é stata la sostituzione dei Raspberry con dispositivi senza SO. Si é scelto quindi di optare per una soluzione basata su micro-chip ATmega328, nello specifico utilizzando delle schede Arduino Nano (Figura 4.1) per sostituire i Pi Zero. Le schede in questione sono completamente prive di sistema operativo e costituite da:

- Micro-chip ATmega328 [41]: microcontrollore CMOS ad architettura RISC ad $8bit$, equipaggiato con memoria flash ISP da $32KB$, $1KB$ di memoria EEPROM e $2KB$ di SRAM. Il chip offre la possibilitá di effettuare operazioni di lettura e scrittura tramite 32 registri general purpose e presenta giá integrati un convertitore A/D a $10bit$ e una linea di comunicazione seriale;

- Tre LED che vengono usati per fornire informazioni sul funzionamento del sistema: due per le linee RX e TX della seriale e uno per indicare l'accensione del dispositivo;
- Un LED built-in accessibile tramite il pin 13;
- Un pulsante di reset per il microcontrollore;
- Interfaccia di comunicazione tramite micro-USB per la programmazione ed eventualmente l'alimentazione;
- 30 pin through-hole che rappresentano i pin del microcontrollore, piú altri per alimentazione e controllo della scheda.

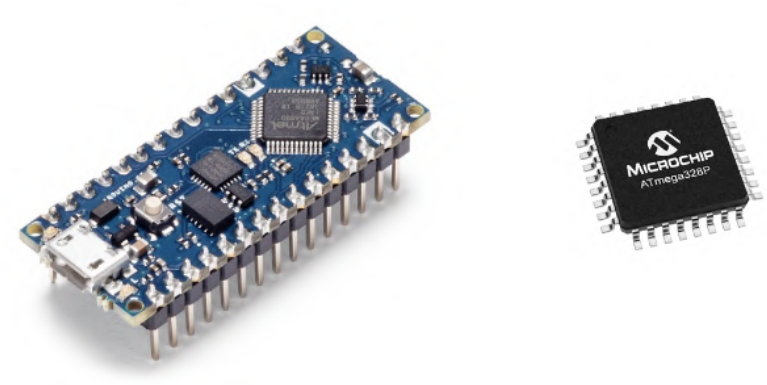


Figura 4.1: A sinistra la scheda PCB Arduino Nano, a destra invece il singolo microcontrollore ATmega328 in configurazione surface-mount

Il microcontrollore su cui si basa la scheda é programmabile collegando il dispositivo ad un computer e usando il programma fornito gratuitamente da Arduino, e accetta codice scritto in linguaggio C o $C++$ (per i nostri scopi verrá usato il primo). Il paradigma di controllo della scheda é, come le caso del Raspberry, il super-loop ed é possibile programmare il dispositivo interagendo direttamente con i pin del microcontrollore. Essendo pensato per applicazioni a bassa potenza é possibile alimentarlo tra gli $1.8V$ e i $5.5V$, e quest'ultimo é anche il valore massimo che la scheda é in grado di fornire in uscita sul singolo pin.
Uno dei motivi per cui si é scelto questa tipologia di dispositivo sta nel fatto che sia possibile utilizzare tre differenti modalitá di risparmio energetico, scegliendo quanto a fondo spegnere il microcontrollore, fino all'opzione ultima in cui viene lasciato acceso soltanto il timer del clock.

La scheda é stata montata su una breadboard insieme ad una serie di componenti addizionali e sensori, atti a realizzare una vera e propria stazione di raccolta dati di tipo:

- Temperatura e umiditá;
- Intensitá luminosa;
- Altitudine e pressione atmosferica;
- Presenza di gas nell'aria.

Per ognuna di queste funzionalitá si é scelto di utilizzare sensori through-hole preassemblati su PCB, per non avere necessitá di effettuare saldature e poter modificare il sistema in corso d'opera. Di seguito saranno presentati i sensori utilizzati, elencandone le caratteristiche e le motivazioni per cui sono stati scelti:

Sensore di temperatura e umiditá. In questo caso si é scelto di utilizzare un DHT11, analogo a quello utilizzato per la prima scheda realizzata con i Raspberry. Le motivazioni sono principalmente legate al costo molto basso. L'alimentazione ideale é a $3.3/3.5V$ ma funziona correttamente fino a $5.5V$.

Sensore di intensitá luminosa. Il sensore di luce ambientale scelto é il BH1750, che si basa sul BH1750FVI IC di ROHM Semiconductor. Si interfaccia tramite il bus I2C alla scheda ed é in grado di leggere il valore in lumen della luminositá ambientale. Il range di valori gestibile va da $0.11lux$ a $65535lux$ e la precisione é di circa $1lux$. Viene alimentato a $3.3V$ e la scheda non ha bisogno di inserire resistenze intermedie per il collegamento poiché sono giá pre-montate sulla stessa.

Sensore di pressione atmosferica. Il sensore scelto per la misura della pressione atmosferica é il BMP280 di Bosch, montato su una scheda PCB di Adafruit che integra anche i componenti per collegamento e alimentazione; anche questo é interfacciabile con interfaccia I2C ed é principalmente un misuratore piezoelettrico di pressione atmosferica con precisione di circa $1.5hPa$. Con La scheda scelta si é in grado tuttavia anche di misurare la temperatura, con precisione di $1°$.

Sensore di gas. Per questo tipo di misura si é scelto un sensore MQ-6, montato su una scheda PCB di JOY-IT, in grado di individuare concentrazioni di gas di propano liquido (LPG) e gas di butano nell'aria. Il sensore

é alimentabile a $5V$ ed é in grado di misurare concentrazioni da $200ppm$ a $10000ppm$. Sul retro della scheda sono integrate la circuiteria necessaria al collegamento ed un potenziometro, usabile per regolare la sensibilitá del pin digitale con cui viene collegato al microcontrollore (non é quindi collegabile su bus I2C).

Insieme a questi quattro sensori (visibili in Figura 4.2) sulla scheda é montata anche un'antenna a radiofrequenza per la comunicazione, uno schermo LCD per la visualizzazione delle informazioni e due LED per monitorare il funzionamento del sistema.

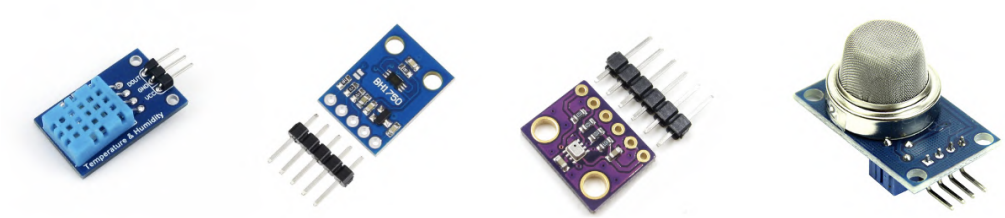


Figura 4.2: Da sinistra: DHT11, BH1750, BMP280, MQ-6

Lo scopo dell'interfaccia a radiofrequenza é quello di sostituire la comunicazione tramite WiFi e quindi ridurre il consumo elettrico del sistema. Per abilitare questo tipo di comunicazione si é scelta una scheda Kuman nRF24L01+ con antenna integrata(Figura 4.3), operante a $2.4GHz$ sulla banda internazionale ISM.

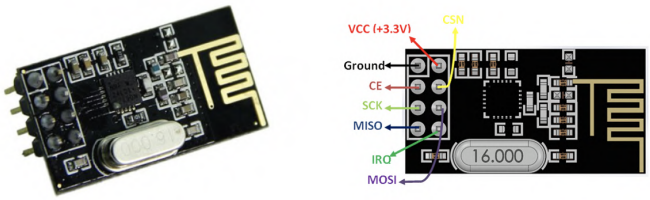


Figura 4.3: Scheda integrata con antenna planare Kuman nRF24L01+ e relativo pinout

Questa antenna comunica su una banda libera e puó essere usata, almeno in linea teorica, fino a distanze pari a $100m$, con baud-rate che vadano da $250kbps$ a $2Mbps$. Particolatiá del dispositivo é la sua capacitá di utilizzare

125 differenti canali, il che apre alla possibilitá di utilizzare 125 antenne all'interno dello stesso ambiente senza problemi di interferenze: i canali sono distribuiti dai $2400MHz$ ai $2525MHz$ ed é possibile su ognuno di questi, comunicare con fino a sei diversi dispositivi. Questo significa che teoricamente un Arduino Nano puó comunicare la stessa informazione sullo stesso canale con fino a sei ricevitori differenti. Il consumo energetico della scheda é molto basso e la corrente che viene richiesta é pari a $12mA$ durante la trasmissione (il che é inferiore a quello del singolo LED per esempio), mentre l'alimentazione é necessariamente tra gli $1.9V$ e i $3.6V$; gli altri pin dell'antenna sono peró in grado di lavorare anche a $5V$, il che ci permette di collegare l'antenna direttamente all'Arduino senza che vi siano convertitori intermedi.

Il protocollo di comunicazione usato dall'antenna verso Arduino é un master-slave Serial Pheriferal Interface (SPI) e pertanto sono necessari quattro pin oltre a quelli di alimentazione e massa:

- Serial Clock (SCK): linea per il clock emesso dal master;
- Master Input Slave Output (MISO): ingresso per il master ed uscita per lo slave;
- Master Output Slave Input (MOSI): uscita del master;
- Chip Select (CS): emesso dal master per definire con quale dispositivo slave vuole comunicare.

I dati sul bus SPI sono trasmessi sfruttando dei registri a scorrimento di dimensione solitamente pari ad $8bit$. Ogni dispositivo master o slave possiede un registro a scorrimento interno i cui bit vengono emessi e, contemporaneamente, immessi, tramite l'uscita MOSI e l'ingresso MISO. Tramite questo registro vengono inviati comandi agli slave e trasmessi dati. Ad ogni impulso di clock i dispositivi che stanno scambiando dati sulle linee del bus emettono un bit dal loro registro interno e lo sostituiscono con un bit fornito dall'altro interlocutore.

I dati campionati dai sensori vengono impacchettati in stringhe di massimo $32byte$ e inviati tramite questa scheda. La scarsa dimensione del pacchetto inviabile ha costretto, in fase di implementazione del codice, ad optare per un invio differenziato di ogni dato singolarmente, piuttosto che per la creazione di un singolo oggetto contenente tutte le misure, come nel caso del Raspberry Pi Zero.

L'altro lato della comunicazione a radiofrequenza é costituito da una scheda analoga, codificata per funzionare come ricevitore, mentre quella sul nodo é utilizzata come trasmettitore. Per effettuare questa configurazione é sufficiente avvalersi delle librerie specifiche della famiglia RF24, tramite le quali é possibile definire degli indirizzi di scrittura e lettura tramite i quali le schede saranno in grado di comunicare. La comunicazione avviene aprendo una "pipe" all'indirizzo scelto, il quale permette di scegliere tra uno dei 125 canali disponibili, e configurando il trasmettitore per inviare dati a tale indirizzo, mentre il ricevitore viene lasciato in ascolto sul canale impostato, in attesa di ricevere un dato. In ottica super-loop il sistema é pensato per campionare ad ogni ciclo un singolo sensore e inviare il dato cosí ottenuto al ricevitore: qualora la ricezione venga confermata con un ACK, viene campionato il dato successivo, altrimenti il sistema ritenta con il dato corrente.
Lo schermo LCD e i LED presenti sul trasmettitore permettono di monitorare lo stato del sistema e comprendere se il funzionamento sia corretto. In Figura 4.4 é mostrato uno dei nodi di raccolta dati implementato con connessione a radiofrequenza.

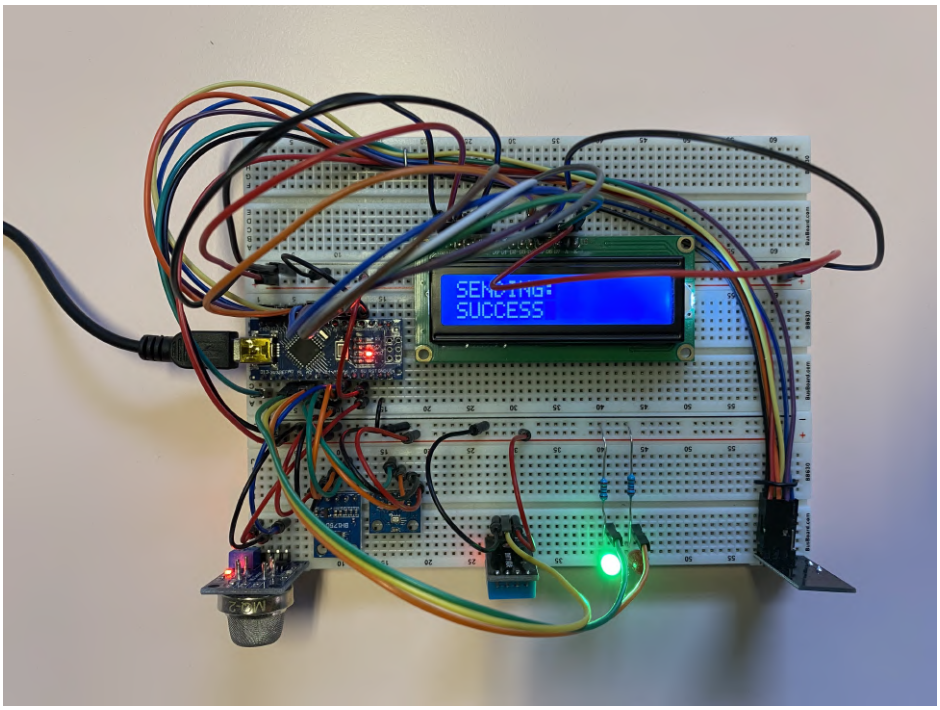


Figura 4.4: Nodo di raccolta dati basato su Arduino Nano equipaggiato con antenna RF (in basso a destra) e una serie di sensori e LED (in basso a sinistra)

4.1.2 Da Raspberry Pi 4 ad Arduino Mega

Come accennato, la parte di ricezione dati é stata realizzata anch'essa con una scheda Kuman nRF24L01+, ma in questo caso non é stato usato un Arduino Nano ma un Arduino Mega. Infatti questo componente deve rimpiazzare il server centrale, originariamente implementato con un Raspberry Pi 4, ed é stato ritenuto di interesse utilizzare una versione di Arduino che permettesse l'integrazione con uno schermo in modo da poter visualizzare i dati in tempo reale, ed eventualmente interagire con il sistema. L'Arduino Mega é quindi equipaggiato con uno schermo da 2.4 pollici TFT TouchScreen, montabile come shield sul dispositivo, andando ad occuparne la maggior parte dei pin, alimentandolo a $3.3V$. Questo schermo (Figura 4.5) é in grado di mostrare tutti i colori RGB e sfrutta il chip XPT2046 per la componente touch-screen. Il touch-screen é di tipo thin-fil (TF) ed é resistivo, quindi la qualitá non risulta eccelsa ed é necessario utilizzare un pennino per una maggiore funzionalitá.

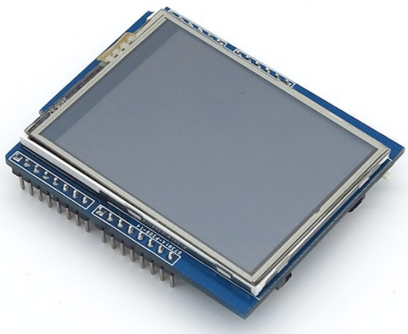


Figura 4.5: Shield touch screen TFT lcd da 2.4" per Arduino Mega

La scheda di ricezione dati riceve quindi pacchetti individuali della lunghezza di $32B$ massimo, corrispondenti nel caso di una stringa a 32 caratteri, e li invia tramite interfaccia seriale ad un computer, che viene anche sfruttato come alimentazione per la scheda. Su quest'ultimo, in questa prima fase, viene mantenuto in esecuzione uno script *Python*, il quale legge da seriale e invia la transazione ricevuta alla *DevNet* di *IOTA* tramite la libreria apposita. É quindi evidente come l'eliminazione della comunicazione WiFi comporti la necessitá di ripensare il meccanismo di invio dati al *Tangle*, poiché questo deve essere necessariamente effettuato tramite interfaccia internet. Sebbene questa prima implementazione sfrutti un computer per questo passaggio non

é difficile integrare, in ottica futura, un modulo GSM sul ricevitore, rendendolo in grado di inviare tramite rete 4G/5G il pacchetto ad un server remoto che si occuperá poi di registrarlo sul *Tangle*.

In Figura 4.6 viene mostrata la scheda di ricezione dati in funzione; nello schermo sono infatti visibili gli ultimi quattro pacchetti ricevuti, corrispondenti ognuno alla lettura di un diverso sensore. É da notare come la presenza di uno shield per il montaggio dello schermo abbia costretto ad adottare una soluzione particolare per l'alimentazione dell'antenna RF: questa ha infatti la necessitá di essere alimentata a $3.3V$, ma questo tipo di uscita é presente sull'Arduino Mega solo nei pin giá occupati dallo shield e non viene replicata in quelli esterni. Per questo motivo, come prima soluzione, non disponendo di soluzioni alternative, si é scelto di aggiungere un Arduino Nano alla breadboard, alimentandolo tramite l'uscita a $5V$ dell'Arduino Mega e usando la sua uscita a $3.3V$ per alimentare l'antenna a RF.

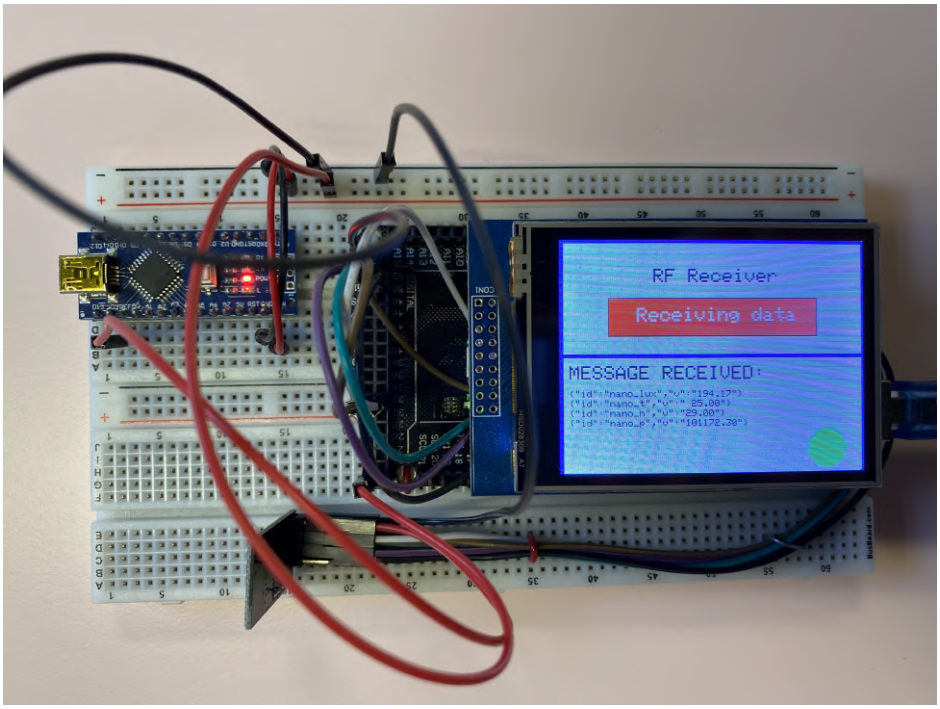


Figura 4.6: Nodo ricevitore basato su Arduino Mega equipaggiato con antenna RF (in basso al centro) e schermo TFT touch-screen

4.1.3 Considerazioni finali e test futuri

Sia per la scheda di ricezione che per quella di trasmissione sono ancora da effettuare approfonditi test sul TPS raggiungibile e sul consumo energetico, ma risulta a priori evidente come questa implementazione sia nettamente piú economica della precedente: infatti, mentre il costo di un Raspberry Pi 4 varia dai 50 euro per la versione piú economica dotata di $4GB$ di RAM, ai 90 euro per la versione dual-core con $8GB$ di RAM, il costo di una scheda Arduino Mega oscilla intorno ai 15 euro. Analogamente la differenza di costo tra un Raspberry Pi Zero e un Arduino Nano é di circa 5 euro il che, sebbene su una singola scheda non sia impattante, risulta importante nel momento in cui si considera una topologia a stella dove siano presenti anche centinaia di schede di raccolta dati.

Per quanto riguarda il consumo energetico invece é necessario ad oggi effettuare ancora una serie di test approfonditi di funzionamento, posizionando i dispositivi in ambiente reale e predisponendo anche un'alimentazione a pannello solare. Alcuni test effettuati precedentemente con un Raspberry Pi 3 senza collegamento di sensori permettono di avere un valore stimato da cui partire che corrisponde a circa $96Wh$ giornalieri; questo non risulta un valore eccessivamente alto se pensiamo al consumo standard di un computer fisso. Questo dispositivo della famiglia Raspberry ha caratteristiche intermedie tra il Pi Zero e il Pi 4 ed é possibile usare i dati raccolti come punto di partenza per i test futuri. Nell'ottica di voler alimentare il dispositivo mediante un pannello solare dobbiamo considerare che la corrente di avviamento richiesta é pari a circa $700 - 900mA$, quindi, usando un pannello standard per sistemi embedded come il MIKROE-651, che eroga $4V$ e $100mA$, sono necessari nove pannelli in parallelo solo per garantirne l'avviamento, supponendo che questi funzionino a regime. Considerando che la dimensione di un pannello é di $70 \times 65mm$ é facile capire come l'alimentazione solare imponga un'opera di integrazione particolarmente impegnativa a livello di superficie occupata, oltre che per il costo. É da considerare inoltre la necessitá in inserire una batteria intermedia per la conservazione dell'energia e un sistema di Maximum Power Point Tracking (MPPT) per garantire il corretto funzionamento del dispositivo. In ottica di alimentarlo solamente a batteria alcuni test effettuati hanno mostrato come una batteria LiPo da 1200mAh sia in grado di garantire alimentazione continua per sole sei ore al dispositivo; questo risultato é chiaramente incompatibile con gli scopi del progetto.
Passando all'uso di un Arduino Nano, anche in questo caso senza considerare il collegamento dei vari sensori, alcuni testi hanno evidenziato consumi di circa $15 - 20Wh$ giornalieri, decisamente inferiori rispetto a quelli individuati

per Raspberry Pi Zero.

I dati qui presentati sono tuttavia puramente rappresentativi poiché non sono stati ancora effettuati test approfonditi, ed é quindi ancora in fase di discussione la modalitá di alimentazione dell'Arduino Nano tramite pannello solare, che risulta comunque decisamente piú abbordabile rispetto a quella del Raspberry Pi Zero. Inserendo una logica funzionamento in risparmio energetico é possibile supporre che il consumo energetico totale giornaliero sia di poco superiore a quello dei sensori nel loro periodo di campionamento, moltiplicato per il numero di dati raccolti al giorno.

Per quanto riguarda invece il TPS, come abbiamo giá dimostrato, il collo di bottiglia si trova a livello di codice ed é rappresentato principalmente dalla latenza presente per la registrazione dei dati sul *Tangle*; a fianco di questo si aggiunge il tempo necessario al dispositivo per campionare i dati e il delay da introdurre tra una misura e l'altra per la comunicazione tramite l'antenna a RF. Non é quindi possibile definire un valore di TPS massimo raggiungibile, ma é comunque da sottilineare come l'implementazione della comunicazione a RF necessariamente causi una diminuzione di questo valore, poiché risulta ora necessario inviare un pacchetto dati per ogni misura e non piú un pacchetto singolo che le contenga tutte, come era possibile tramite interfaccia WiFi. Questo viene comunque compensato dal risparmio energetico elevato che se ne ottiene, dalla possibilitá di alimentare il dispositivo tramite pannello fotovoltaico e sopratutto dalla diminuzione del costo e l'aumento della scalabilitá del sistema.

4.2 Sostituzione della blockchain Hyperledger Fabric

Come é stato giá precedentemente evidenziato l'uso di una *blockchain Hyperledger* non rappresenta la soluzione migliore per realizzare un paradigma *BCoT*, principalmente a causa della difficoltá di gestire la proprietá dell'informazione e l'astrazione del *digital-twin*. Questo secondo concetto indica la rappresentazione tramite un asset su *blockchain*, o in generale un asset digitalizzato, di un oggetto fisico o di una parte di esso. Le potenzialitá offerte dal concetto di *digital-twin* sono molto semplici da comprendere, e fanno tutte capo alla possibilitá di gestire digitalmente l'intera vita di un oggetto, aggiornandone lo stato, la proprietá e le caratteristiche. L'idea del *digital-twin* puó essere declinata nei piú differenti campi di applicazione, rappresentando

oggetti reali come anche documenti, informazioni o qualsiasi tipo di "media"; la caratteristica comune di tutte queste implementazioni é la necessitá di scambiare l'oggetto garantendone la non alterazione, mantenendo traccia sia dello stato iniziale che di quello corrente.

In questa sezione verrá presentato un lavoro ancora in corso d'opera, che sostituisce la *blockchain* realizzata con *Hyperledger Fabric* con una implementata tramite *bigchainDB*, presentando le motivazioni concettuali e cercando di mostrare anche l'effettivo workflow nello scambio di un oggetto generico tra due utenti.

4.2.1 L'uso di bigchainDB per costruire digital-twin

Dallo studio effettuato su varie implementazioni *blockchain*, ricercando le caratteristiche principali per la realizzazione dell'astrazione *digital-twin*, é emerso come *bigchainDB* possa essere la tecnologia piú promettenente per alcune sue caratteristiche specifiche:

1. *BigchainDB* é in grado di creare e gestire blocchi sulla catena che siano di tipo *owner-controlled*, cioé che rappresentino asset specifici legati alla chiave pubblica dell'utente che attualmente li possiede. Questo, unito al fatto che il campo *asset* all'interno della transazione permetta di inserire un qualsiasi oggetto JSON, abilita la possibilitá di creare copie digitali di un oggetto e rappresentarle sulla *blockchain* mediante transazioni legate al proprietario dell'oggetto stesso;

2. La *blockchain* suddivide automaticamente le transazioni in due categorie: "SPENT" e "UNSPENT", differenziando quindi quelle che un proprietario ha giá scambiato (le prime) da quelle che invece non ha ancora scambiato (le seconde). Con questa logica, unita al fatto che é possibile etichettare transazioni come *CREATE* o *TRANSFER*, é possibile mantenere traccia della creazione di un *digital-twin* e di tutti gli scambi che avvengono, con la possibilitá di percorrerne la storia a ritroso attraverso le varie *TRANSFER* fino alla *CREATE*, per garantirne la non alterazione;

3. Costituisce ultima caratteristica che ne favorisce l'adozione la facilitá di utilizzo e di implementazione di *bigchainDB*. Infatti, nonostante sia una tecnologia completa e "ready-to-market", *bigchainDB* offre gratuitamente a chiunque voglia utilizzarlo un'immagine Docker pronta per essere installata su un computer, utilizzabile per implementare un

nodo in maniera immediata. Vengono poi fornite diverse librerie, definiti "driver" (tra le quali si é scelto di usare quella *Javascript*) per interfacciarsi con il nodo, che espone nativamente un'interfaccia API contattabile tramite chiamate HTTP.

Unico punto debole di *bigchainDB*, ma che risulta una mancanza comune di questo tipo di sistemi open-source (presente per esempio anche in *Hyperledger Fabric* e *IOTA*) é la scarsa documentazione, che risulta a volte addirittura errata e costringe ad effettuare numerosi test autonomi per comprendere tutte le funzionalitá disponibili.

4.2.2 Un caso studio pratico

Un esempio di caso di studio molto interessante é quello del monitoraggio strutturale: pensiamo per esempio ad un'infrastruttura pubblica, come un ponte, e consideriamo il caso in cui sia necessario monitorare lo stato di deterioramento dello stesso per valutare quando effettuare interventi di manutenzione o simili. Questo esempio ci porta a reintrodurre l'idea della *BCoT* come elemento di monitoraggio e raccolta dati, ma ci costringe anche ad integrare una gestione documentale relativa. Supponiamo quindi che ogni asset, con il proprio *digital-twin*, sia rappresentativo di un modulo di raccolta dati, comprendente vari sensori, il cui scopo sia quello di monitorare una parte del ponte e raccogliere differenti informazioni. Il nodo registrerá questi dati sul *ledger* come transazioni ma anche lo stato stesso dell'asset sará costantemente monitorato per verificare che il sistema funzioni correttamente e che ne sia effettuata l'effettiva manutenzione. A questo, in un caso studio reale, si aggiunge la necessitá di digitalizzare anche tutta la parte documentale relativa ai vari interventi, e sopratutto quella relativa alla costruzione e alla manutenzione del ponte.
Ogni utente che produce un documento viene quindi registrato su una *blockchain* realizzata tramite *bigchainDB*, assegnandogli una coppia chiave pubblica/chiave privata e registrando la sua identitá su un DB a parte, in modo che l'utente sia in grado di gestire *owner-controlled-assets* e che sia sempre possibile collegare la chiave pubblica all'utente che la possiede. Il sistema puó essere utilizzato per garantire la responsabilitá, anche legale, rispetto alle informazioni legate ad una chiave pubblica, e per garantire la possibilitá di trasferire documenti e informazioni tra utenti, registrandone lo scambio di proprietá e associando a questo anche il passaggio della responsabilitá riguardo al determinato documento. La *blockchain* in tutto questo viene utilizzata

per certificare e rendere accessibile il documento che viene trasferito, mantenendo nella transazione iniziale di *CREATE* la hash del documento originale e garantendone cosí l'immutabilitá.

Sulla *blockchain* sono quindi presenti quattro tipologie di transazioni:

- Transazioni che creano *digital-twin* per documenti: sono transazioni di *CREATE* che contengono l'informazione riguardante un documento, corredate da informazioni sullo stesso e da allegati; queste identificano l'autore del documento o il responsabile attuale;
- Transazioni che creano *digital-twin* per le stazioni di raccolta dati: sono transazioni di *CREATE* che creano un asset relativo al sensore, assegnandogli peró una coppia chiave pubblica/chiave privata come se fosse un utente, in modo che i dati registrati dal sensore siano recuperabili facilmente;
- Transazioni che trasferiscono *digital-twin*: sono transazioni di *TRANSFER* pensate solo per gli asset che rappresentano documenti, utilizzabili per condividere informazioni e trasferire la responsabilitá relativa ad un determinato documento ad una persona terza;
- Transazioni che registrano dati: sono transazioni che si riferiscono ad un determinato nodo, legate alla sua chiave pubblica, e che registrano semplicemente dati sul *ledger*.

Il sistema che viene presentato in questa sezione é stato realizzato tramite un'immagine *bigchainDB* installata tramite Docker su un server Linux, per implementare un singolo nodo. Il DB utilizzato per mantenere invece le informazioni sugli utenti é *CouchDB*, simile a *MongoDB*, giá nominato in precedenza. A questa parte realizzata tramite *bigchainDB* viene affiancato l'interfacciamento con il *Tangle* della *DevNet* di *IOTA*, il cui uso sará motivato successivamente. L'interfacciamento dell'utente con il sistema avviene tramite una dashboard realizzata tramite *HTML/CSS* e con back-end in *Javascript*.

É di particolare importanza sottolineare come, vista la necessitá di mantenere l'identitá degli utenti che si interfacciano al sistema su un DB a parte, che ne contenga anche l'associazione tra le chiavi e l'utente, sono state utilizzati due strumenti di cifratura per rendere le informazioni sicure. Per prima cosa la password dell'utente viene cifrata mediante la libreria *bcrypt*, che si basa sull'omonima funzione, e rappresenta l'algoritmo di hashing di password

di default per molte distribuzioni Linux [42]. In secondo luogo anche per la chiave privata viene effettuato un processo di cifratura, che si basa sulla funzione *aes-256-ctr*, della classe *Advanced Encryption Standard (AES)* [43], che sfrutta un cifrario a blocchi di tipo *Counter (CTR)*; questo tipo di cifratura viene reso possibile dalla libreria *Javascript* di nome *Crypto* e la funzione di cifratura richiede di inserire una password sulla quale questa verrá effettuata, e che sará poi usata per decifrare il valore.
La differente scelta di implementazione deriva da necessitá pratiche: é infatti sufficiente, per la password di accesso al DB, confrontare la password inserita, criptata dal sistema, con quella criptata salvata sul DB; qualora siano identiche significa che la password inserita é corretta e questa non viene mai mostrata o trasmessa in chiaro al server. Per la chiave privata é invece necessario che l'utente possa recuperarla in chiaro, per usarla nel sistema al momento della firma delle transazioni, per cui questa viene salvata sul DB cifrata e viene poi decifrata lato client al bisogno.

4.2.3 Esempio di flusso di lavoro

Un esempio di flusso di lavoro potrebbe essere il seguente:

Sia l'utente $ALICE$, registrato al sistema e associato con una coppia chiave pubblica/chiave privata, il responsabile della costruzione di una struttura. Per la registrazione all'utente viene chiesto nome e cognome, l'azienda a cui appartiene (se ne ha una), il codice fiscale (usato per identificare univocamente l'utente), una password per l'accesso e una differente (chiamata *encryption key*) per cifrare e decifrare la chiave privata. Su *CouchDB* viene quindi aggiunto un file JSON appartenente al DB "users", la cui struttura é riportata in Figura 4.7.

Essendo il primo utente registrato sulla *blockchain*, $ALICE$ viene considerata il responsabile del progetto, e colei che assegna i compiti ai vari attori successivi e ne controlla l'operato. Per esempio BOB potrebbe essere la persona adibita all'installazione di una serie di sensori lungo la struttura; anche BOB quindi sarebbe rappresentato su *CouchDB* esattamente come $ALICE$, ovviamente con una chiave privata e pubblica differente. $ALICE$ utilizza quindi la sua chiave privata per firmare una transazione che contiene un documento in cui viene assegnato a BOB questo lavoro e, per completare l'assegnazione, effettua una *TRANSFER* con con tale documento; in Figura 4.8 é mostrato il blocco che rappresenta il documento su *CouchDB*, mentre in Figura 4.9 é mostrato lo stesso documento come asset sulla *blockchain* (come ci si aspetterebbe il blocco porta la chiave pubblica di $ALICE$). Infine, in

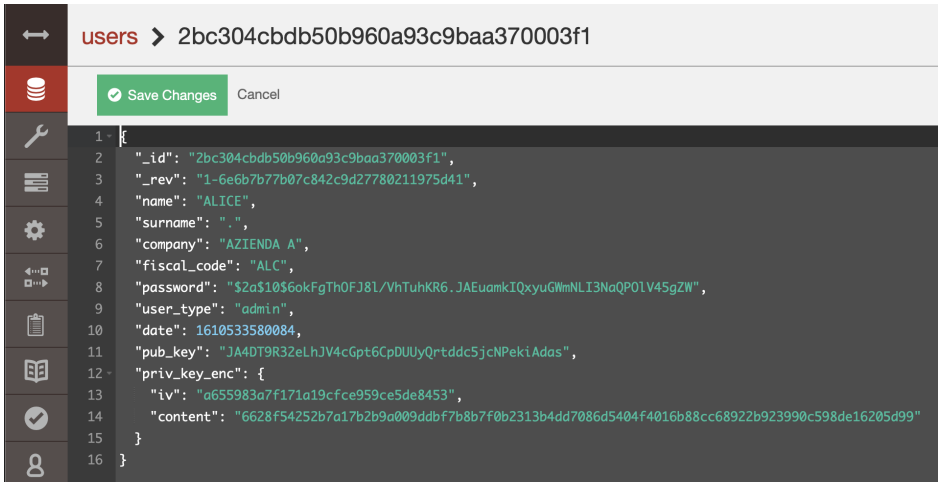


Figura 4.7: Oggetto JSON che rappresenta ALICE su couchDB

Figura 4.10 viene riportato un esempio di visualizzazione delle due transazioni con cui il documento é stato creato e trasferito, dove viene registrato il momento del trasferimento da *ALICE* a *BOB* ed é altresí indicato l'ID della transazione di *TRANSFER* mostrata in Figura 4.11.
Ricevuto il documento, *BOB* lo firma per ritrasferirlo ad *ALICE*, garantendo quindi di aver accettato il lavoro e la relativa responsabilitá, e viene generata la transazione mostrata in Figura 4.12. *BOB* utilizza poi la sua chiave privata per registrare sul *ledger* una serie di transazioni che rappresentino ogni sensore, assegnando anche ad ognuno una chiave pubblica e una privata, in modo che possano registrare autonomamente dati sul *ledger*. I sensori saranno quindi autonomamente in grado di usare la loro chiave privata per firmare blocchi che rappresentano i valori misurati, e che saranno ricercabili sul *ledger* grazie alla chiave pubblica specifica di ognuno.
Supponiamo poi che *CHARLIE* sia invece delegato da *ALICE* alla gestione documentale, occupandosi di ottenere i permessi e le autorizzazioni per la costruzione; anche in questo caso *CHARLIE* possiede una chiave privata con cui firma le transazioni che inserisce nel *ledger* e puó, qualora fosse necessario, trasferire gli asset ad *ALICE* affinché questa li firmi con la sua chiave privata per garantire di averli visualizzati e approvati.

In un sistema cosí realizzato é possibile garantire in ogni momento quale utente abbia la responsabilitá relativa ad un determinato documento, osservando la catena di trasferimento di proprietá del blocco specifico; é possibile inoltre garantire che il blocco non sia stato manomesso, visto che nella transazione *CREATE* iniziale é registrato il documento originale con la sua hash.

L'unico problema rimasto é quello di garantire la trasparenza delle informazioni contenute nel *ledger*. L'idea di usare una *blockchain* costrutita con *bigchainDB* costringe a lavorare necessariamente con *blockchain private*, poiché i nodi sono solitamente nell'ordine delle poche unitá (al massimo una decina) e il massimo livello di decentralizzazione ottenibile é quello nel quale siano distribuiti tra i vari attori che vi si interfacciano, supponendo ognuno sia parte di una differente azienda. Questo livello di decentralizzazione é relativamente scarso e non garantisce, soprattutto in caso di controversie legali, che le aziende non possano accordarsi per distruggere l'intera rete e quindi il *ledger*, nascondendo comportamenti poco virtuosi.
La risposta a questo rimane l'uso del sistema ibrido mostrato in precedenza, dove *IOTA* viene comunque utilizzata per registrare una copia di ogni transazione che venga effettuata su *bigchainDB*, sia essa una *CREATE* o una *TRANSFER*.

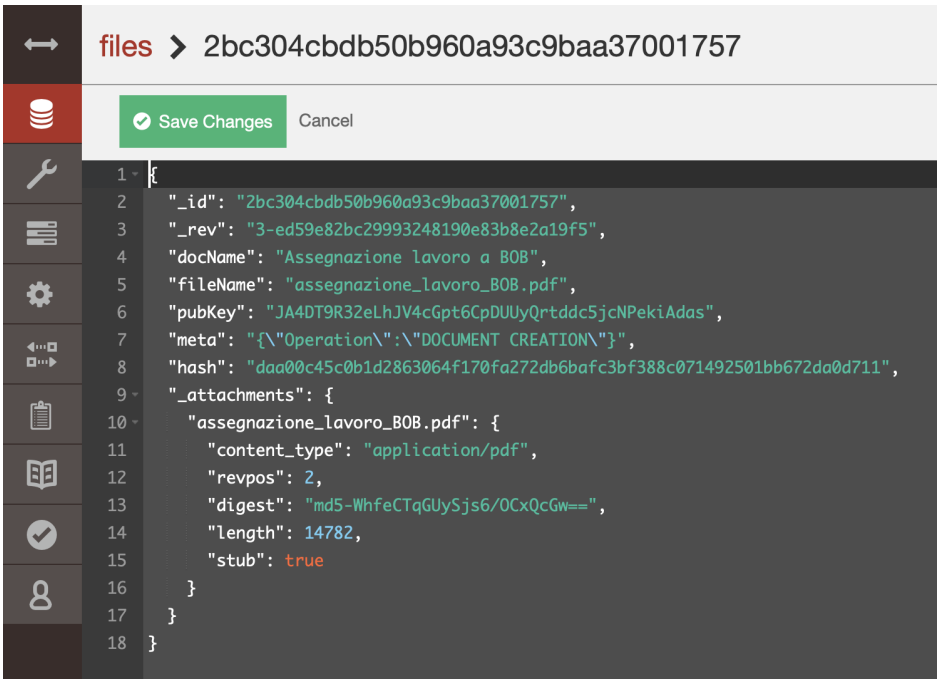
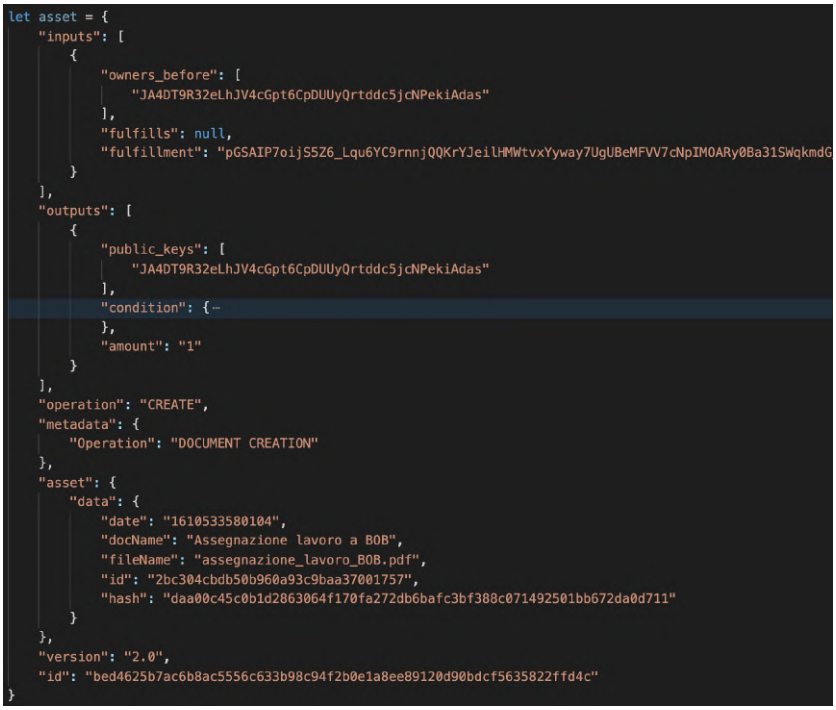


Figura 4.8: Oggetto JSON che rappresenta il documento caricato con la sua hash



```
let asset = {
    "inputs": [
        {
            "owners_before": [
                "JA4DT9R32eLhJV4cGpt6CpDUUyQrtddc5jcNPekiAdas"
            ],
            "fulfills": null,
            "fulfillment": "pGSAIP7oijS5Z6_Lqu6YC9rnnjQQKrYJeilHMWtvxYyway7UgUBeMFVV7cNpIMOARy0Ba31SWqkmdG
        }
    ],
    "outputs": [
        {
            "public_keys": [
                "JA4DT9R32eLhJV4cGpt6CpDUUyQrtddc5jcNPekiAdas"
            ],
            "condition": {-
            },
            "amount": "1"
        }
    ],
    "operation": "CREATE",
    "metadata": {
        "Operation": "DOCUMENT CREATION"
    },
    "asset": {
        "data": {
            "date": "1610533580104",
            "docName": "Assegnazione lavoro a BOB",
            "fileName": "assegnazione_lavoro_BOB.pdf",
            "id": "2bc304cbdb50b960a93c9baa37001757",
            "hash": "daa00c45c0b1d2863064f170fa272db6bafc3bf388c071492501bb672da0d711"
        }
    },
    "version": "2.0",
    "id": "bed4625b7ac6b8ac5556c633b98c94f2b0e1a8ee89120d90bdcf5635822ffd4c"
}
```


Figura 4.9: Oggetto JSON che rappresenta l'asset sulla blockchain

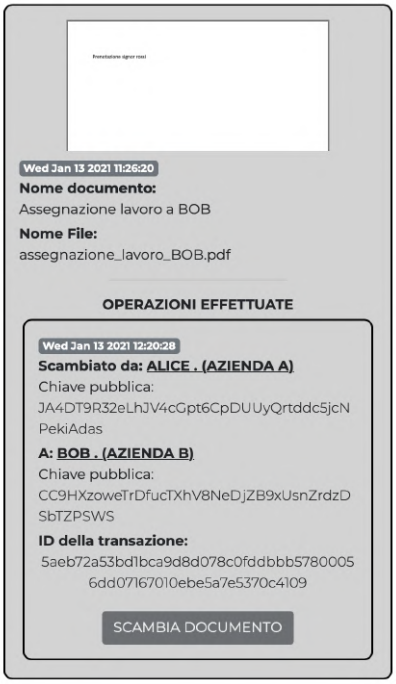
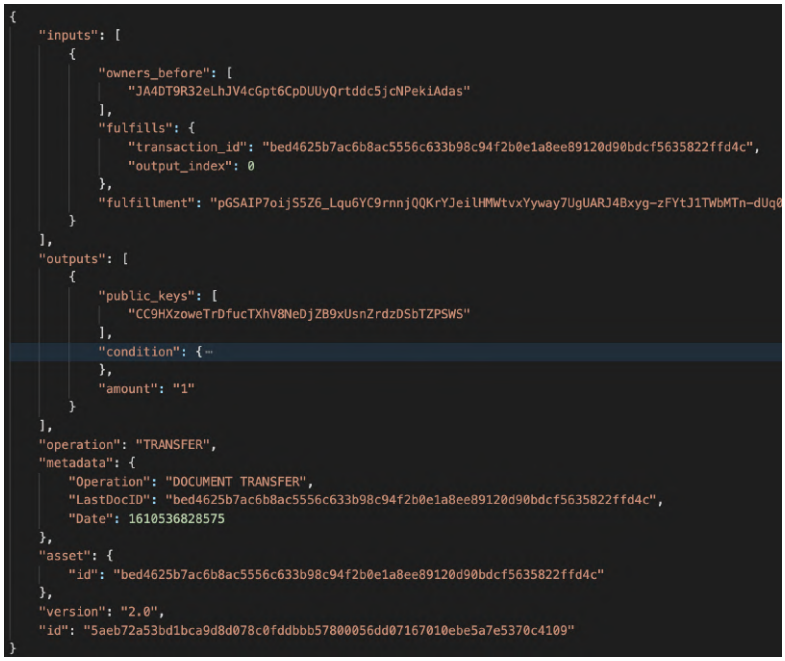
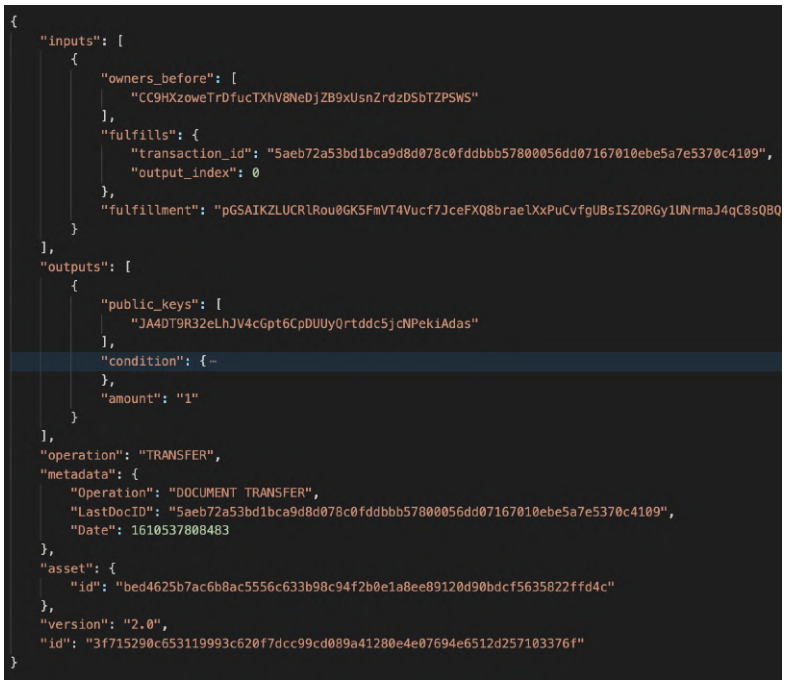


Figura 4.10: Visualizzazione grafica della transazione tra ALICE e BOB.



```
{
    "inputs": [
        {
            "owners_before": [
                "JA4DT9R32eLhJV4cGpt6CpDUUyQrtddc5jcNPekiAdas"
            ],
            "fulfills": {
                "transaction_id": "bed4625b7ac6b8ac5556c633b98c94f2b0e1a8ee89120d90bdcf5635822ffd4c",
                "output_index": 0
            },
            "fulfillment": "pGSAIP7oijS5Z6_Lqu6YC9rnnjQQKrYJeilHMWtvxYyway7UgUARJ4Bxyg-zFYtJ1TWbMTn-dUq0
        }
    ],
    "outputs": [
        {
            "public_keys": [
                "CC9HXzoweTrDfucTXhV8NeDjZB9xUsnZrdzDSbTZPSWS"
            ],
            "condition": {…
            },
            "amount": "1"
        }
    ],
    "operation": "TRANSFER",
    "metadata": {
        "Operation": "DOCUMENT TRANSFER",
        "LastDocID": "bed4625b7ac6b8ac5556c633b98c94f2b0e1a8ee89120d90bdcf5635822ffd4c",
        "Date": 1610536828575
    },
    "asset": {
        "id": "bed4625b7ac6b8ac5556c633b98c94f2b0e1a8ee89120d90bdcf5635822ffd4c"
    },
    "version": "2.0",
    "id": "5aeb72a53bd1bca9d8d078c0fddbbb57800056dd07167010ebe5a7e5370c4109"
}
```


Figura 4.11: Transazione con cui ALICE scambia il documento con BOB.



```
{
    "inputs": [
        {
            "owners_before": [
                "CC9HXzoweTrDfucTXhV8NeDjZB9xUsnZrdzDSbTZPSWS"
            ],
            "fulfills": {
                "transaction_id": "5aeb72a53bd1bca9d8d078c0fddbbb57800056dd07167010ebe5a7e5370c4109",
                "output_index": 0
            },
            "fulfillment": "pGSAIKZLUCRlRou0GK5FmVT4Vucf7JceFXQ8braelXxPuCvfgUBsISZORGy1UNrmaJ4qC8sQBQ
        }
    ],
    "outputs": [
        {
            "public_keys": [
                "JA4DT9R32eLhJV4cGpt6CpDUUyQrtddc5jcNPekiAdas"
            ],
            "condition": {…
            },
            "amount": "1"
        }
    ],
    "operation": "TRANSFER",
    "metadata": {
        "Operation": "DOCUMENT TRANSFER",
        "LastDocID": "5aeb72a53bd1bca9d8d078c0fddbbb57800056dd07167010ebe5a7e5370c4109",
        "Date": 1610537808483
    },
    "asset": {
        "id": "bed4625b7ac6b8ac5556c633b98c94f2b0e1a8ee89120d90bdcf5635822ffd4c"
    },
    "version": "2.0",
    "id": "3f715290c653119993c620f7dcc99cd089a41280e4e07694e6512d257103376f"
}
```


Figura 4.12: Transazione con cui BOB reinvia il documento ad ALICE.

Capitolo 5

Conclusioni e ringraziamenti

In conclusione, all'interno di questo elaborato si é cercato di presentare una visione tecnica ma anche relativamente personale del paradigma *BCoT*. Questo é stato necessario poiché la tecnologia *blockchain* risulta ad oggi ancora troppo giovane, ed il panorama internazionale é ricco di differenti correnti di pensiero a riguardo. Da un lato troviamo i piú conservatori, sostenitori delle prime implementazioni di *blockchain* e convinti che ogni lavoro che se ne discosti sia soltanto una bolla temporanea pronta ad esplodere; dall'altro troviamo invece i piú "progressisti", tra i quali mi colloco, convinti che continuare a lavorare su strutture rigide non possa portare alcun beneficio allo sviluppo e all'adozione della *blockchain* stessa, e che vi sia sicuramente spazio per nuovi campi di applicazione e nuove tecnologie all'interno del panorama. Lo scopo ultimo che ci si era posti é stato certamente raggiunto, proponendo una spiegazione tecnica precisa delle tecnologie presenti, selezionandone una per ogni categoria, e mostrando come queste possano essere integrate con sistemi *IoT* per fornire all'industria soluzioni semplici, scalabili e sopratutto accessibili per risolvere alcune criticitá emergenti e non. Nei lavori futuri sará necessario investigare la fattibilitá dell'applicazione su larga scala delle soluzioni proposte, nell'ottica di mantenere sempre alto il livello di personalizzazione del progetto, al fine di declinarlo nel modo migliore per ogni caso studio che verrá proposto.

In merito a questo é certamente da sottolineare come l'ultimo capitolo di questo elaborato sia frutto del lavoro della startup BLOCKvision, i cui primi passi risalgono al febbraio 2019, e che dai laboratori di Telecomunicazioni dell'Universitá di Bologna mi ha portato nell'ultimo anno a confrontarmi con il mercato, con l'essere imprenditore e con il lavorare a stretto contatto con grandi realtá e importanti aziende. Senza la presenza della startup questa sarebbe stata solo una tesi di progetto, mentre é ora a tutti gli effetti il punto

di partenza per costruire un'azienda e un futuro di successo, cosa che l'Universitá mi ha certamente aiutato a creare.

In tutto questo quindi é doveroso nominare il mio primo partner nel progetto, un amico prima che un collega, Sergiu Popescu, che da quei laboratori per primo ha creduto con me in quest'attivitá e al quale devo gran parte dei progressi tecnologici che l'azienda ha fatto fino ad oggi.

Accanto a lui, separati solo perché entrati successivamente a far parte del progetto, devo ringraziare anche Bruno Venditti, Reem Emilia Mansour e Vito Sardone, che da Giugno 2020. Dopo averli incontrati all'evento finale di StartUp Day 2020, sono entrati a far parte del progetto e hanno lavorato come se l'idea fosse stata loro dall'inizio, contribuendo a portare il progetto al punto in cui ora si trova, e diventando anche loro ottimi amici oltre che colleghi.

Un ringraziamento va anche all'incubatore CesenaLab, che ci ha appoggiato dall'inizio di questo progetto e ha creduto in noi, fornendoci spazio, consigli e supporto costante negli ultimi mesi, aiutandoci a costruire il nostro progetto.

Infine un ringraziamento speciale va anche a Martina, che mi ha supportato in questi anni di universitá e mi ha convinto ad intraprendere questo percorso.

Elenco delle figure

Bibliografia

[1] Satoshi Nakamoto, (2008). Bitcoin: A Peer-to-Peer Electronic Cash System,

[2] Secp256k1 curve [Internet]

`https://en.bitcoin.it/wiki/Secp256k1`

[3] La Byzantine Fault Tolerance spiegata [Internet]

`https://academy.binance.com/it/articles/byzantine-fault-tolerance-explained`

[4] [Internet]

`https://www.electroyou.it/m_dalpra/wiki/byzantine-fault-tolerance-la-chiave-per-la-blockchain`

[5] Wouter Penard, Tim van Werkhoven. On the Secure Hash Algorithm family

[6] Protocol documentation [Internet]

`https://en.bitcoin.it/wiki/Protocol_documentation#Hashes`

[7] Linux Foundation. An Introduction to Hyperledger

[8] [Internet]

`https://medium.com/coinmonks/how-does-hyperledger-fabric-works-cdb68e6066f5`

[9] Raft consensus algorithm [Internet]

https://raft.github.io

[10] Diego Ongaro and John Ousterhout, Stanford University. In Search of an Understandable Consensus Algorithm,

[11] BigchainDB GmbH, (2018). BigchainDB 2.0: The Blockchain Database

[12] Jae Kwon, (2014). Tendermint: Consensus without Mining

[13] [Internet]

https://blog.bigchaindb.com/bigchaindb-2-0-is-byzantine-fault-tolerant-5ffdac96bc44

[14] Cosmos: A Network of Distributed Ledgers [Internet]

https://cosmos.network

[15] Getting started with IOTA [Internet]

https://docs.iota.org/docs/getting-started/1.1/introduction/overview

[16] [Internet]

https://docs.iota.org/docs/getting-started/1.1/introduction/architecture

[17] The Tangle, Sergey Popov, 2018

[18] [Internet]

https://blog.iota.org/the-tangle-an-illustrated-introduction-c0a86f994445/

[19] Joshua S. Speagle, (2020). A Conceptual Introduction to Markov Chain Monte Carlo Methods

[20] Illustrated introduction to Tangle [Internet]

https://blog.iota.org/the-tangle-an-illustrated-introduction-79f537b0a455/

[21] Iota Roadmap [Internet]

`https://roadmap.iota.org`

[22] Iota Coordicide [Internet]

`https://coordicide.iota.org`

[23] Iota Foundation, (2020). The Coordicide

[24] IOTA Proof of Work [Internet]

`https://medium.com/bytes-io/iota-proof-of-work-remote-vs-local-explained-1cbd89392a79`

[25] IOTA Tutorial 2, Trit and Trytes, IOTA News [Internet]

`https://iota-news.com/iota-tutorial-trit-and-tryte/`

[26] Morris J. Dworkin, (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

[27] Sponge function [Internet]

`https://keccak.team/sponge_duplex.html`

[28] [Internet]

`https://medium.com/bytes-io/iota-proof-of-work-remote-vs-local-explained-1cbd89392a79`

[29] About accounts, IOTA Documentation [Internet]

`https://docs.iota.org/docs/getting-started/1.1/accounts/overview`

[30] [Internet]

`https://docs.iota.org/docs/getting-started/1.1/cryptography/addresses`

[31] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hulsing, Markus Ruckert. On the Security of the Winternitz One-Time Signature Scheme

[32] Perkins, D.N., Brune Drisse, M.N., Nxele, T., Sly, P.D., (2014). E- Waste: A Global Hazard. Annals of Global Health, Volume 80, Issue 4, Pages 286-295, ISSN 2214-9996

[33] European Parliamentary Research Service (EPRS), (2015). Understanding waste streams Treatment of specific waste

[34] Centro di Coordinamen to RAEE (Cdc RAEE), Il trattamento dei RAEE 2015 (2015)

[35] Ecodom Sustainability Report 2018 [Internet]

`http://www.ecodom- consorzio.it/it/sostenibilita`

[36] Reshma Kamath Northwestern University, Chicago, IL, USA (2018). Food Traceability on Blockchain: Walmart's Pork and Mango Pilots with IBM

[37] From shore to plate: Tracking tuna on the blockchain, Provenance [Internet]

`https://www.provenance.org/tracking-tuna-on-the-blockchain#overview`

[38] TradeLens blockchain-enabled digital shipping platform continues expansion with addition of major ocean carriers Hapag-Lloyd and Ocean Network Express, Maersk [Internet]

`https://www.maersk.com/news/articles/2019/07/02/hapag-lloyd-and-ocean-network-express-join-tradelens`

[39] Iota Tangle Explorer [Internet]

`https://explorer.iota.org/devnet`

[40] CouchDB documentation [Internet]

`https://docs.couchdb.org/en/stable/intro/overview.html`

[41] ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet

[42] Provos, Niels; Mazieres, David; Talan Jason Sutton 2012 (1999). A Future-Adaptable Password Scheme. Proceedings of 1999 USENIX Annual Technical Conference: 81-92.

[43] Announcing the ADVANCED ENCRYPTION STANDARD (AES), in Federal Information Processing Standards Publication 197, United States National Institute of Standards and Technology (NIST), 26 novembre 2001.

[44] Lipmaa, Helger; Wagner, David; Rogaway, Phillip (2000). "Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption".