

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



CORSO DI LAUREA IN INFORMATICA

TESI DI LAUREA TRIENNALE IN

INFORMATICA

**Roaming telefonico di nuova generazione con
Hyperledger Fabric**

Relatore:

Prof. Christiancarmine Esposito

Correlatore:

Gino Ciccone

Candidato:

Luigi Vollono

Matr. 05121/05597

Anno Accademico 2019/2020

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
Via Giovanni Paolo II, 132, 84084 Fisciano SA

"Le idee, come le persone, possono fiorire solo quando sono libere."

Cit. Charles Robert Darwin

Ringraziamenti

Molte persone hanno contribuito alla mia crescita, non solo professionale.

Un primo ringraziamento va ai miei genitori per il supporto che mi è sempre stato dato, i quali sono sempre stati presenti e sempre lo saranno.

Un altro ringraziamento va ai miei amici che già conoscevo da tempo, ovvero Danilo e Alessandro, coi quali ho preparato i vari esami assieme a loro, e colleghi di università che ho conosciuto durante il percorso universitario, in particolare al mio gruppo di studio composto da Francesco, Gianandrea, Vincenzo, Davide, Martina e molti altri, dove abbiamo condiviso momenti divertenti tra una lezione e l'altra.

E un ultimo ringraziamento va ai professori universitari che ho incontrato in questi anni, i quali mi hanno insegnato molto, come ad essere sempre pronto e professionale, ma soprattutto a metterci la passione in quel che si fa e si studia.

Abstract

Il roaming consiste nel consentire agli utenti di telefonia cellulare di usufruire di servizi di altri operatori, quando non si ha copertura da parte del proprio operatore di appartenenza. Sebbene sia sfruttato da decenni, le attuali soluzioni di roaming dovranno affrontare nuove sfide imposte dalla transizione verso i sistemi 5G, ovvero il supporto a comunicazioni veloci, scalabili e sicure in grado di soddisfare i requisiti delle nuove applicazioni di Internet of Things, Smart City o Industrial 4.0. I tradizionali sistemi di roaming sono implementati mediante un pattern di comunicazione diretto tra operatori, o mediante opportuni intermediari mentre nei sistemi 5G sarà necessario un modello di interazione più scalabile e decentralizzato. Per poter realizzare tale modello architetturale, la blockchain è una tecnologia appetibile. Essa si realizza come una struttura di dati organizzati come blocchi collegati tra loro mediante l'applicazione retroattiva di funzioni crittografiche di hashing, e l'inserimento di nuovi blocchi in maniera decentralizzata applicando un opportuno algoritmo di consenso distribuito tra i partecipanti alla blockchain. L'uso dell'hashing garantisce l'immutabilità dei dati una volta che la loro immissione è stata approvata, e il consenso distribuito implementa una soluzione di consistenza debole tra repliche dei dati distribuiti su scala geografica.

Nel contesto del presente lavoro di tesi si è studiato l'applicazione della blockchain, implementando le operazioni CRUD (Create, Read, Update, Delete) di gestione dei dati di roaming sottoforma di smart contract, ed articolando la gestione di dati di roaming mediante la generazione di transazioni nella blockchain. Nello specifico è stata implementata una applicazione sfruttando la piattaforma open source Hyperledger Fabric, che realizza una blockchain privata

con permessi. In tale piattaforma, i nodi partecipanti sono noti tra loro, identificati e dotati di apposite autorizzazioni, realizzando una rete affidabile e controllata.

La scelta di tale piattaforma è stata motivata dall'esigenza di proteggere il roaming da possibili attacchi esterni. La soluzione definita implementa un'architettura detta federated, in quanto, gli operatori coinvolti fanno parte della stessa rete blockchain. Quando un cliente, abbonato ai servizi di un operatore telefonico, effettua il roaming, l'operatore della rete ospitante genera una istanza con le informazioni dell'abbonato, la incapsula in una transazione e la sottoscrive alla blockchain a cui partecipa. A valle dell'esecuzione del consenso, tutti i partecipanti alla blockchain avranno presso il proprio nodo locale una copia dell'istanza suddetta.

La tesi inizia introducendo il sistema di roaming attuale, elencandone le principali problematiche. Successivamente, è presentata l'integrazione della tecnologia blockchain nei sistemi di roaming, illustrandone vantaggi e caratteristiche. Viene introdotto e approfondito Hyperledger Fabric, descrivendo le principali caratteristiche che lo differenziano dalle altre piattaforme blockchain. A seguire, viene specificato il progetto e l'implementazione del sistema proposto, illustrando lo scenario applicativo simulato. Infine, viene illustrata l'esecuzione del sistema proposto, illustrando alcune metriche di qualità ottenute empiricamente. La tesi è conclusa presentando possibili sviluppi futuri, come l'interoperabilità tra blockchain, elemento chiave per poter realizzare un'architettura not federated in cui gruppi di operatori adottano tecnologie blockchain differenti e devono poter interoperare tra loro.

INDICE

1. INTRODUZIONE	1
1.1 DESCRIZIONE DEL PROBLEMA	1
1.2 SOLUZIONE PROPOSTA	2
1.3 PANORAMICA DELLA TESI.....	3
2. GESTIONE DEL ROAMING	4
2.1 ESEMPIO DI ROAMING.....	4
2.2 COSA È IL ROAMING	6
2.3 SERVIZI DI ROAMING	6
2.4 SISTEMA DI ROAMING ATTUALE	9
2.4.1 CLEARING HOUSE	11
2.5 FUTURO DEL ROAMING	13
2.6 PROBLEMATICHE DEL ROAMING.....	14
3. BLOCKCHAIN-BASED ROAMING.....	17
3.1 PERCHÉ USARE BLOCKCHAIN	17
3.2 COSA È UNA BLOCKCHAIN	19
3.3 CARATTERISTICHE PRINCIPALI DI UNA BLOCKCHAIN.....	21
3.4 STRUTTURA DI UNA BLOCKCHAIN	22
3.4.1 BLOCKCHAIN PUBBLICA (O SENZA AUTORIZZAZIONE)	23
3.4.2 BLOCKCHAIN PRIVATA (O CON AUTORIZZAZIONE)	24
3.4.3 DISTRIBUTED LEDGER.....	25
3.4.4 CONSENSO.....	26
3.4.5 CRITTOGRAFIA	27
3.5 SMART CONTRACT.....	28
3.6 INTEROPERABILITÀ TRA BLOCKCHAIN	28
3.6.1 SOLUZIONI DI INTEROPERABILITÀ ESISTENTI	29
3.6.2 PROPRIETÀ DELLE SOLUZIONI DI INTEROPERABILITÀ.....	32
4. HYPERLEDGER FABRIC	33
4.1 COSA È HYPERLEDGER FABRIC.....	34
4.1.1 UN NUOVO APPROCCIO	37
4.1.2 PRIVACY E RISERVATEZZA	38

4.2 RETE BLOCKCHAIN IN FABRIC	39
4.3 IDENTITÀ	42
4.3.1 PUBLIC KEY INFRASTRUCTURE (PKI)	42
4.3.2 CERTIFICATI DIGITALI	43
4.3.3 CERTIFICATION AUTHORITY (CA)	44
4.4 MEMBERSHIP SERVICE PROVIDER (MSP)	45
4.5 POLITICHE	46
4.6 I PEER	47
4.7 LEDGER	48
4.7.1 WORLD STATE	49
4.7.2 BLOCKCHAIN (DEL LEDGER)	51
4.7.3 BLOCCHI	52
4.7.4 TRANSAZIONI	53
4.8 ORDERING SERVICE	55
4.8.1 FLUSSO DELLE TRANSAZIONI	55
4.9 SMART CONTRACT (O CHAINCODE)	57
5. SISTEMA PROPOSTO	59
5.1 SCENARIO	59
5.2 INSTALLAZIONE	60
5.3 IMPLEMENTAZIONE	61
6. SISTEMA IN ESECUZIONE	69
6.1 ESECUZIONE	69
6.2 VALUTAZIONE	73
6.3 MIGLIORAMENTI FUTURI	76
6.4 CONCLUSIONE	76
BIBLIOGRAFIA	78

INDICE DELLE FIGURE

Figura 1. Operazione di roaming	5
Figura 2. Scenario di roaming: servizio voce	7
Figura 3. Scenario di roaming: servizio dati	8
Figura 4. Processo di fatturazione in una chiamata	9
Figura 5. Relazione diretta	11
Figura 6. Relazione tramite un terzo operatore	11
Figura 7. Relazione tramite Clearinghouse	12
Figura 8. Rappresentazione generica di un sistema blockchain	21
Figura 9. Struttura di una blockchain.....	23
Figura 10. Tipologie di soluzione di interoperabilità tra blockchain	31
Figura 11. Panoramica dell'architettura di rete di Fabric.....	40
Figura 12. Funzionamento di una PKI	43
Figura 13. Rappresentazione di peer contenenti smart contract e ledger	47
Figura 14. Struttura di un ledger in Fabric.....	49
Figura 15. Struttura del world state	50
Figura 16. Struttura della blockchain del ledger.....	52
Figura 17. Struttura di un blocco	53
Figura 18. Struttura di una transazione	54
Figura 19. Chaincode contenenti Smart Contract	58
Figura 20. Metodo per inizializzare il ledger.....	61
Figura 21. Metodo per creare un nuovo asset	62
Figura 22. Metodo per leggere un asset all'interno del ledger	62
Figura 23. Metodo per aggiornare un asset presente nel ledger	63
Figura 24. Metodo per eliminare un asset dal ledger	63
Figura 25. Metodo che verifica se esiste un dato asset nel ledger.....	64
Figura 26. Metodo che restituisce tutti gli asset contenuti nel ledger	64
Figura 27. Registrazione amministratore dell'applicazione	65
Figura 28. Registrazione utente dell'applicazione.....	66
Figura 29. Preparazione della connessione al canale e allo smart contract	66

Figura 30. Operazioni effettuate dalla prima applicazione	67
Figura 31. Operazioni effettuate dalla seconda applicazione	67
Figura 32. Avvio della rete e creazione canale	70
Figura 33. Distribuzione del chaincode sulla rete.....	70
Figura 34. Esecuzione della prima applicazione	71
Figura 35. Esecuzione della seconda applicazione	72
Figura 36. Grafico tempo di latenza delle transazioni interne	73
Figura 37. Grafico tempo di latenza di insiemi di transazioni	74
Figura 38. Grafico tempo di latenza di Sawtooth	75

CAPITOLO 1

1. INTRODUZIONE

In questo capitolo viene fornita una panoramica generale della tesi.

Nella prima sezione viene esaminato il problema che si intende risolvere attraverso questo lavoro di tesi.

Nella seconda sezione viene menzionata la soluzione proposta.

Infine, nella terza sezione viene illustrata la struttura della tesi, definendo i capitoli e di cosa trattano.

1.1 DESCRIZIONE DEL PROBLEMA

Il roaming consiste nel consentire agli utenti di telefonia cellulare di usufruire di servizi di altri operatori, quando non si ha copertura da parte del proprio operatore di appartenenza. Sebbene sia sfruttato da decenni, le attuali soluzioni di roaming dovranno affrontare nuove sfide imposte dalla transizione verso i sistemi 5G, ovvero il supporto a comunicazioni veloci, scalabili e sicure in grado di soddisfare i requisiti delle nuove applicazioni di Internet of Things, Smart City o Industrial 4.0. I tradizionali sistemi di roaming sono implementati mediante un pattern di comunicazione diretto tra operatori, o mediante opportuni intermediari mentre nei sistemi 5G sarà necessario un modello di interazione più scalabile e decentralizzato. Per poter realizzare tale modello architetturale, la blockchain è una tecnologia appetibile. Essa si realizza come una struttura di dati organizzati come blocchi collegati tra loro mediante l'applicazione retroattiva di funzioni

crittografiche di hashing, e l'inserimento di nuovi blocchi in maniera decentralizzata applicando un opportuno algoritmo di consenso distribuito tra i partecipanti alla blockchain. L'uso dell'hashing garantisce l'immutabilità dei dati una volta che la loro immissione è stata approvata, e il consenso distribuito implementa una soluzione di consistenza debole tra repliche dei dati distribuiti su scala geografica.

1.2 SOLUZIONE PROPOSTA

Nel contesto del presente lavoro di tesi si è studiato l'applicazione della blockchain, implementando le operazioni CRUD (Create, Read, Update, Delete) di gestione dei dati di roaming sottoforma di smart contract, ed articolando la gestione di dati di roaming mediante la generazione di transazioni nella blockchain. Nello specifico è stata implementata una applicazione sfruttando la piattaforma open source Hyperledger Fabric, che realizza una blockchain privata con permessi. In tale piattaforma, i nodi partecipanti sono noti tra loro, identificati e dotati di apposite autorizzazioni, realizzando una rete affidabile e controllata. La scelta di tale piattaforma è stata motivata dall'esigenza di proteggere il roaming da possibili attacchi esterni. La soluzione definita implementa un'architettura detta federated, in quanto, gli operatori coinvolti fanno parte della stessa rete blockchain. Quando un cliente, abbonato ai servizi di un operatore telefonico, effettua il roaming, l'operatore della rete ospitante genera una istanza con le informazioni dell'abbonato, la incapsula in una transazione e la sottoscrive alla blockchain a cui partecipa. A valle dell'esecuzione del consenso, tutti i partecipanti alla blockchain avranno presso il proprio nodo locale una copia dell'istanza suddetta.

1.3 PANORAMICA DELLA TESI

La tesi inizia introducendo il sistema di roaming attuale, elencandone le principali problematiche. Successivamente, è presentata l'integrazione della tecnologia blockchain nei sistemi di roaming, illustrandone vantaggi e caratteristiche. Viene introdotto e approfondito Hyperledger Fabric, descrivendo le principali caratteristiche che lo differenziano dalle altre piattaforme blockchain. A seguire, viene specificato il progetto e l'implementazione del sistema proposto, illustrando lo scenario applicativo simulato. Infine, viene illustrata l'esecuzione del sistema proposto, illustrando alcune metriche di qualità ottenute empiricamente. La tesi è conclusa presentando possibili sviluppi futuri, come l'interoperabilità tra blockchain, elemento chiave per poter realizzare un'architettura not federated in cui gruppi di operatori adottano tecnologie blockchain differenti e devono poter interoperare tra loro.

CAPITOLO 2

2. GESTIONE DEL ROAMING

In questo capitolo viene introdotto cosa è e come viene gestito il roaming oggi, enunciando anche le sue problematiche.

Nella prima sezione viene illustrato un esempio di roaming, con l'obiettivo di dare una idea generale di come funziona.

Nella seconda sezione si entra nel dettaglio, spiegando cosa è effettivamente il roaming e gli attori che ne fanno parte.

Nella terza sezione vengono menzionati i servizi di roaming che è possibile effettuare.

Nella quarta sezione viene illustrato come funziona il sistema di roaming attuale e come interagiscono tra loro la rete domestica con quella ospitate tramite terze parti, ovvero la clearing house, e viene chiarito il ruolo di quest'ultima nella sottosezione dedicata.

Nella quinta sezione viene solo introdotto un possibile utilizzo futuro del roaming per fornire supporto alle reti 5G di nuova generazione.

Infine, nella sesta sezione vengono elencate le problematiche principali che affliggono il sistema di roaming attuale.

2.1 ESEMPIO DI ROAMING

Il roaming viene utilizzato in particolare dagli operatori telefonici di telefonia cellulare per permettere agli utenti mobili di collegarsi tra loro utilizzando

eventualmente anche una rete non di loro proprietà dietro una quota di pagamento all'altro operatore. In altre parole, il roaming è un servizio che consente agli utenti mobili di continuare ad utilizzare il proprio dispositivo mobile per effettuare e ricevere chiamate, SMS, navigare in internet e inviare o ricevere e-mail mentre si è al di fuori della copertura di rete dell'operatore di appartenenza. Questo servizio è consentito da un accordo di roaming tra l'operatore domestico, chiamato Home Network Operator (HNO), di un utente mobile e la rete dell'operatore ospitante, chiamata Visited Network Operator (VNO). Questo accordo riguarda le componenti tecniche e commerciali necessarie per abilitare il servizio.

Quando un utente si trova in un'area non coperta dal suo operatore e accende il proprio dispositivo mobile, quest'ultimo tenta di comunicare con una rete ospitante. La rete ospitante rileva la connessione dal dispositivo, verifica se è registrata nel proprio sistema e tenta di identificare la rete domestica di tale utente. Se esiste un accordo di roaming tra la rete domestica e una delle reti ospitanti, il servizio viene fornito, ad esempio l'inizializzazione di una chiamata.

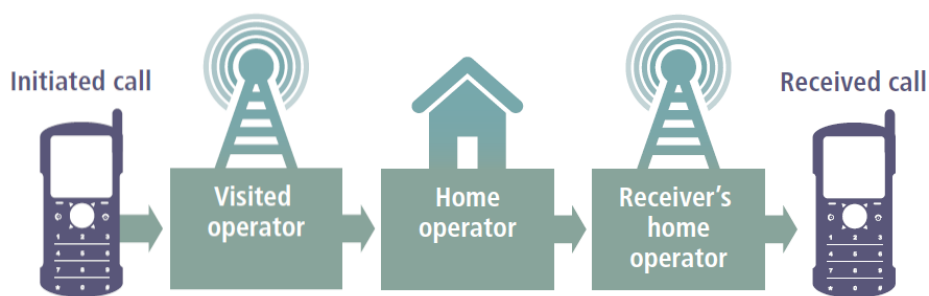


Figura 1. Operazione di roaming

La rete ospitante richiede anche informazioni di servizio dalla rete domestica sull'utente, ad esempio, se il telefono in uso è stato smarrito o rubato e se il dispositivo mobile è autorizzato per l'uso in tale rete.

2.2 COSA È IL ROAMING

Il roaming è la capacità degli abbonati di una rete mobile, denominata HPMN (Home Public Mobile Network), di utilizzare a distanza i servizi di tale rete accedendovi attraverso una rete diversa, denominata VPMN (Visited Public Mobile Network) o semplicemente rete ospitante [1].

Tre attori principali intervengono negli scenari di roaming:

- L'abbonato (o subscriber), che si avvale dei servizi di telecomunicazione forniti;
- l'HPMN, che gestisce l'abbonamento e i servizi dell'utente;
- Il VPMN, nella cui area di copertura geografica gli utenti accedono ai servizi contrattati con l'HPMN.

Per consentire agli abbonati dell'operatore di utilizzare servizi di roaming in una data VPMN, è necessario negoziare preventivamente un accordo di roaming tra i due operatori di telecomunicazioni. Le procedure di questo aspetto aziendale sono normalmente standardizzate, come nel caso del servizio Global System for Mobile Communications (GSM) attraverso le procedure GSM Association.

2.3 SERVIZI DI ROAMING

Il roaming può essere attivato sia per i servizi voce che per i dati [1].

Nel primo caso, il VPMN, prima di fornire l'accesso agli utenti in visita (attraverso il Mobile Switching Center/Visited Location Register [MSC/VLR], che fornisce la copertura geografica agli abbonati richiedenti) interroga l'HPMN

sui servizi a cui questi utenti sono sottoscritti (informazioni contenute nel Home Location Register [HLR], database di proprietà dell'HPMN). Quindi, se gli abbonamenti sono corretti, gli abbonati saranno abilitati ad accedere ai servizi corrispondenti (esempio, la creazione di chiamate vocali).

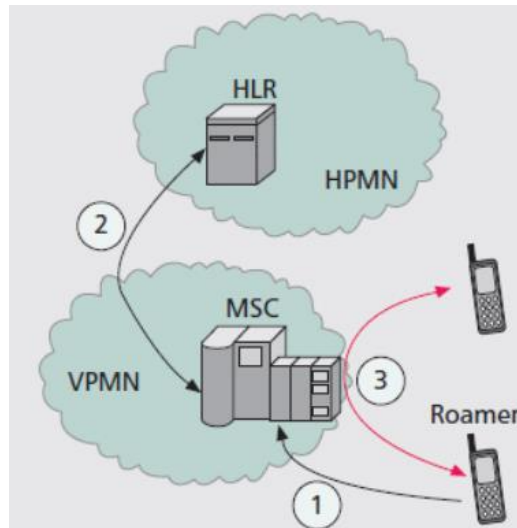


Figura 2. Scenario di roaming: servizio voce

Il roaming dei dati è simile, anche se alcune caratteristiche lo differenziano dal caso delle chiamate vocali. Qui, l'abbonato è associato (dopo aver interrogato l'HLR, database di proprietà dell'HPMN) con un nodo di supporto (chiamato SGSN) appartenente al General Packet Radio Service (GPRS). Quindi il roamer indica la rete dati a cui deve essere effettuata una connessione e viene stabilito un contesto tra di loro tramite un nodo chiamato Gateway GPRS Support Node (GGSN).

2. GESTIONE DEL ROAMING

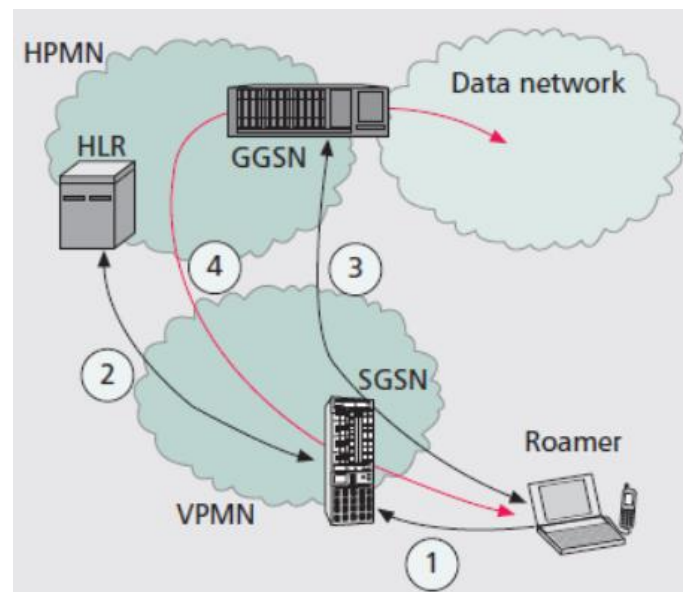


Figura 3. Scenario di roaming: servizio dati

Notare che SGSN appartiene al VPMN, mentre il GGSN si trova nell'HPMN. Pertanto, sia i dati trasmessi che quelli ricevuti dall'abbonato mobile passano necessariamente attraverso l'HPMN. Al contrario, questo non accade nel caso del traffico vocale, per il quale l'interazione VPMN-HPMN è spesso ridotta alla sola query iniziale all'HLR.

2.4 SISTEMA DI ROAMING ATTUALE

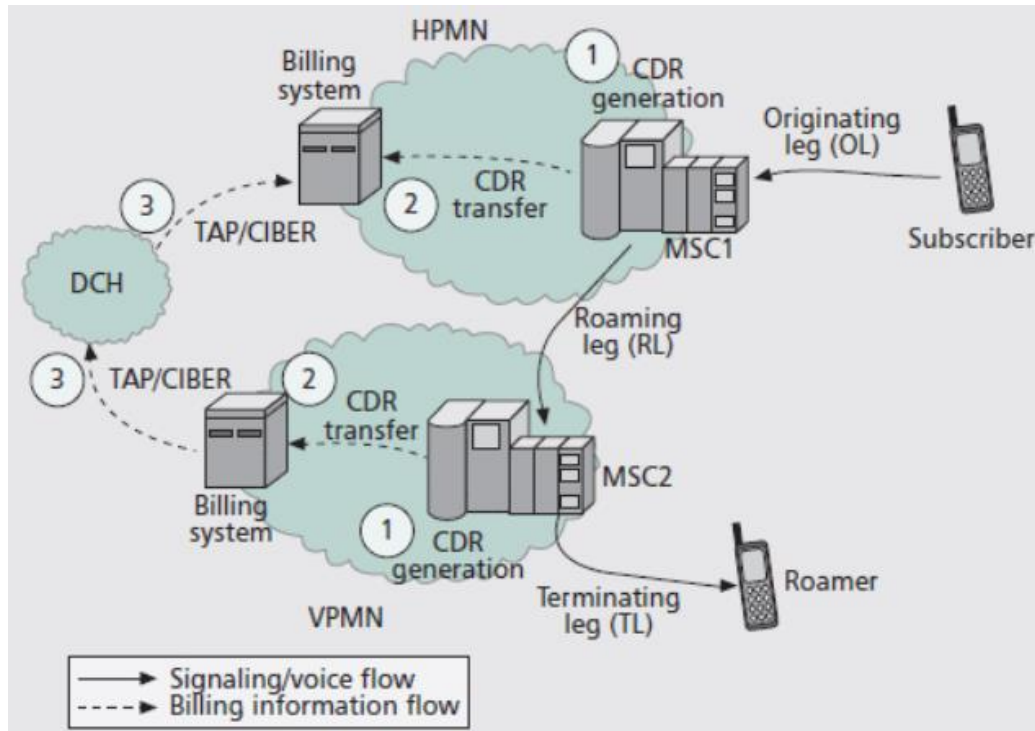


Figura 4. Processo di fatturazione in una chiamata

La Figura 4 mostra uno scenario in cui un abbonato (o subscriber) in un HPMN chiama un roamer in un VPMN.

La chiamata si articola in tre parti, chiamate anche “legs” [1]:

- Un originating leg (OL) tra l’abbonato e MSC1;
- Un roaming leg (RL) tra MSC1 e MSC2;
- Un terminating leg (TL) tra MSC2 e il roamer

L’abbonato in questo scenario verrà addebitato per l’OL, mentre il roamer pagherà per RL e TL. Il processo per addebitare sia l’abbonato che il roamer è il seguente:

- **Fase 1:**

I dati relativi alla chiamata (durata, ora, origine, destinazione, ecc.) vengono raccolti da MSC (o SGSN e GGSN) in file di dati denominati Call Detail Records (CDR) (passaggio 1 in *Figura 4*).

- **Fase 2:**

Successivamente, questi CDR vengono inviati ai sistemi di fatturazione (o billing system) nelle rispettive reti (passaggio 2 in *Figura 4*).

- **Fase 3:**

Questi billing system si occupano dell'elaborazione dei CDR e della generazione delle fatture agli abbonati. Poiché l'HPMN è responsabile della generazione delle fatture sia per l'abbonato che per il roamer, il VPMN invia le informazioni contenute nella CDR all'HPMN (passaggio 3 in *Figura 4*) compilate in una struttura dati ben consolidata. Per questo processo di compilazione, il GSMA ha definito la standard Transfer Account Procedure (TAP), mentre le reti CDMA (code-division multiple access) utilizzano il record CIBER (cellular intercarrier billing exchange roamer). Ci sono tempi ben definiti per il trasferimento di questi file. Per sollevare gli operatori con un gran numero di accordi di roaming dall'oneroso compito di gestire i file TAP / CIBER per ogni operatore ai sensi dell'accordo, alcune società fungono da clearinghouse (o centri di smistamento) per questi dati. Tali società sono chiamate Data Clearing House (DCH). Un DCH è una singola interfaccia per un operatore ed è incaricato di gestire tutti gli aspetti della trasmissione, ricezione e conversione dei file TAP / CIBER per conto dell'operatore che assume il servizio. Infine, una volta ricevuti i file TAP / CIBER, l'HPMN deve pagare il debito specificato da contratto con il VPMN in conformità con le tariffe del contratto di roaming corrispondenti, chiamate interoperator tariffs (IOT).

2.4.1 CLEARING HOUSE

Ogni operatore deve collaborare con molte reti per soddisfare le esigenze di roaming dei propri abbonati, ciò significa che gli operatori devono gestire l'interconnessione a livello globale e scambiare dati con più formati [2].

Dal punto di vista commerciale, devono gestire complicate relazioni finanziarie con svariate leggi e regolamenti in diverse parti del mondo.

Possono esserci vari tipi di relazioni tra operatori per fornire i servizi di roaming:

- Relazione diretta, è un'opzione valida per gli operatori nei paesi vicini in cui è previsto un traffico roaming intenso.

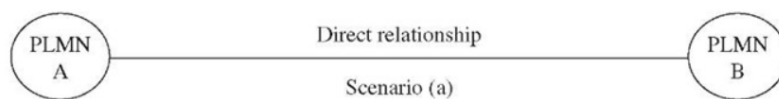


Figura 5. Relazione diretta

- Relazione piggyback, viene utilizzato tra operatori che sono membri della stessa alleanza o gestiti dallo stesso gruppo.

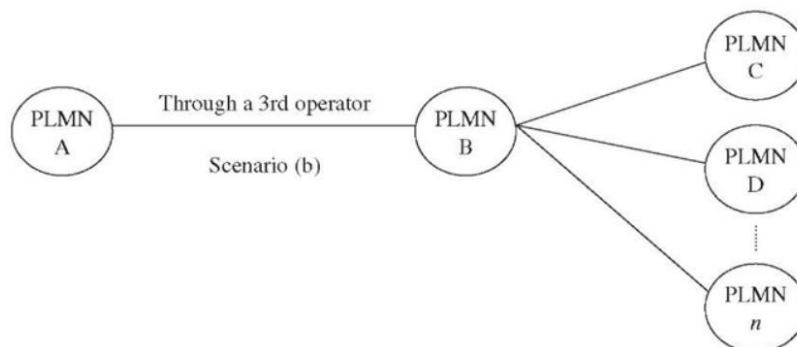


Figura 6. Relazione tramite un terzo operatore

- **Clearing house** svolgono un ruolo importante, ovvero supportare gli operatori nella gestione delle loro operazioni di roaming. Per i nuovi operatori di rete, le clearing house li aiuta ad accelerare l'ingresso nel mercato perché forniscono diversi servizi utili.

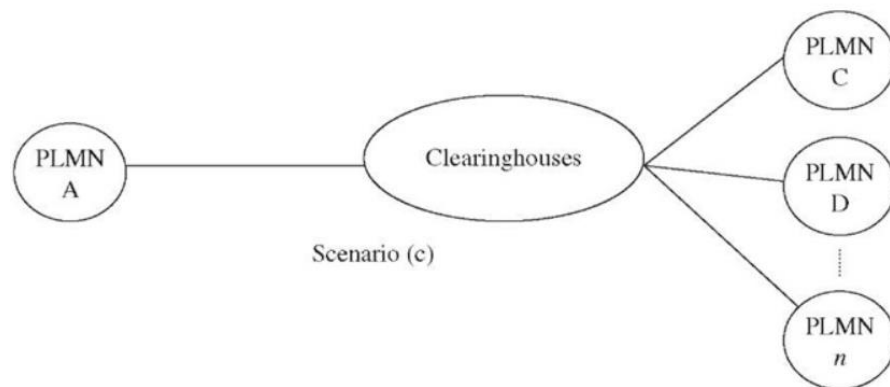


Figura 7. Relazione tramite Clearinghouse

In generale, le clearing house offrono i seguenti servizi o sottoinsiemi di servizi:

- Gestisce gli accordi di roaming: si assume la piena responsabilità della negoziazione, preparazione e documentazione di un accordo di roaming per conto di un PLMN (Public Land Mobile Network);
- Gestisce i partner di roaming: funge da unico punto di contatto per la gestione di più partner di roaming;
- Facilita il roaming interstandard: ciò include, ad esempio, la gestione di diversi formati di fatturazione e pagamento, standard di segnalazione, incompatibilità di switch e di altri problemi tecnologici;
- Compensa e regola i dati: fornisce un processo centralizzato per lo scambio di dati, il regolamento finanziario mensile, la consegna elettronica dei dati, la gestione di IOT e valuta, ecc.;

- Assistenza clienti: funge da fonte per l'attività di roaming. Ad esempio, con i dati di cui dispongono, possono facilmente creare profili di roaming contenenti da dove provengono, quanto tempo rimangono e quali servizi utilizzano. Si tratta di informazioni estremamente utili per l'assistenza clienti e il marketing.
- Gestione delle frodi: aiuta gli operatori a rilevare tempestivamente le frodi. I tipi di frode come la clonazione e l'abbonamento possono essere gestiti in modo efficace. Vengono inviati rapporti sull'utilizzo elevato a HPMN per combattere le frodi in roaming. Il VPMN può implementare autonomamente un sistema di gestione delle frodi o affidare questa funzione alla clearing house per adempiere ai propri obblighi.

2.5 FUTURO DEL ROAMING

Il Local 5G Operator (L5GO) è una delle innovazioni di spicco nel 5G che opera come una rete mobile su piccola scala all'interno di un'area geografica limitata come università, ospedale, fabbrica o centro commerciale [3]. Le L5GO consentono alle entità aziendali di gestire il proprio ecosistema di comunicazione 5G con requisiti distintivi e personalizzati. Il decentramento e la progettazione orientata alla località degli L5GO assicurano alta affidabilità, consapevolezza del contesto, sicurezza perimetrale e gestione della privacy.

Le procedure di roaming e offload mantengono la connettività persistente degli abbonati su diverse reti e aree geografiche. Gli L5GO sono in grado di fornire connettività ai clienti dell'operatore di rete mobile (MNO) quando risiedono al di fuori dell'area di copertura geografica della sua rete domestica. L'offload consente agli MNO di trasferire il carico del traffico di rete ad altre reti per aumentare

l'efficienza di rete del sistema, ridurre al minimo il consumo di energia delle stazioni base, raggiungere la qualità del servizio prevista e massimizzare il throughput. Poiché gli L5GO offrono una copertura migliore, gli MNO possono utilizzare questi L5GO per servire i propri abbonati quando risiedono nell'area di copertura di L5GO. Inoltre, i processi di roaming mancano di trasparenza, il che si traduce nella violazione di accordi e preaccordi statici da parte degli operatori di rete. Inoltre, esiste la possibilità che un operatore partner acceda alle informazioni degli utenti in modo illegale e addebiti ingiustamente agli utenti in roaming. Alla fine, i clienti non sarebbero stati soddisfatti del servizio. Pertanto, la fiducia reciproca tra gli operatori deve essere mantenuta. Inoltre, le caratteristiche della rete in tempo reale come il carico di corrente e la larghezza di banda non vengono valutate in tali accordi. Inoltre, la popolarità degli L5GO attiverà un numero sempre maggiore di istanze di roaming e offload, il che aumenta il carico del traffico di rete negli operatori di rete. Tuttavia, questi problemi dovrebbero essere risolti nella futura evoluzione del 5G.

2.6 PROBLEMATICHE DEL ROAMING

Il roaming offre sia sfide che opportunità per gli operatori, inerenti alle implicazioni commerciali dell'introduzione del roaming regionale, nazionale e internazionale per i servizi voce e dati [4]. Sebbene esistano già diverse soluzioni rimangono ancora problemi critici associati al roaming, come:

- **Fiducia:**

L'accesso illegale alle informazioni relative all'utente in roaming non è affrontato nelle architetture esistenti, che è considerata una grave violazione

della legge sulla protezione dei dati degli utenti. Precisamente, l'operatore di rete della rete ospitante, ovvero il VNO (Visitor Network Operator), può avere politiche diverse da quelle accettate dagli utenti del loro operatore di rete, ovvero l'HNO (Home Network Operator). Ciò richiede l'accettazione dei termini sulla privacy del roaming prima di fornire qualsiasi servizio.

- **Sicurezza:**

L'utente in roaming deve essere autenticato dal VNO tramite l'HNO prima di ottenere l'accesso. Attraverso questa procedura, agli utenti illegali viene impedito l'accesso alla rete. In effetti, ci sono molte autenticazioni utente proposte e protocolli di scambio di chiavi per i servizi di roaming, tuttavia, questi protocolli si concentrano solo sull'autenticazione reciproca tra l'utente in roaming e il VNO. Infatti, gli attuali metodi di roaming non forniscono un buon modo per rilevare e prevenire le frodi. Ad esempio, se qualcuno ruba una scheda SIM da un dispositivo per effettuare chiamate o trasmettere dati, la rete ospitante consentirà la transazione anche se si tratta di una frode. Nel momento in cui la rete domestica rileva l'attività sospetta e la interrompe, le tariffe di roaming per il gestore potrebbero lo stesso accumularsi.

- **Scalabilità:**

Esiste un interesse crescente da parte degli operatori e degli organi di governo per trovare modi per condividere le risorse in modo efficiente. Tuttavia, l'accordo di servizio peer-to-peer esercitato dagli operatori mobili è piuttosto inefficiente. Gli operatori dedicano molto tempo e risorse alla definizione di contratti di roaming con centinaia di partner. Una volta impostato l'accordo, spendono risorse per configurare e testare la connessione con il partner di roaming.

- **Terze parti:**

Ma il problema principale sono le Data Clearing House, cioè gli intermediari e i processi attuali aumentano i tempi di consegna delle transazioni in roaming. Un simile tempo di una transazione in roaming può variare da un paio di giorni a pochi mesi. A causa di questa variazione e dei ritardi, gli operatori devono assumere persone aggiuntive il cui compito è assicurarsi che le transazioni in roaming siano tutte contabilizzate e completate in modo tempestivo, ma questo è solo uno spreco di risorse. In effetti, le tariffe delle clearing house per piccole transazioni stanno portando molti operatori più piccoli a rinegoziare con le Data Clearing House, poiché le tariffe attuali hanno la capacità di ridurre significativamente il loro profitto. Questo problema peggiorerà solo a causa del prezzo graduale del volume delle transazioni impiegato dalle Data Clearing House.

CAPITOLO 3

3. BLOCKCHAIN-BASED ROAMING

In questo capitolo viene analizzata l'integrazione della tecnologia blockchain nei sistemi di roaming e spiegata tale tecnologia.

Nella prima sezione viene spiegato il perché utilizzare un approccio blockchain per affrontare le problematiche del roaming attuale.

Nella seconda sezione viene spiegato il concetto di blockchain in generale.

Nella terza sezione vengono enunciate le caratteristiche principali e i vantaggi di un sistema blockchain.

Nella quarta sezione viene illustrato come è strutturato un tipico sistema blockchain, e spiegate cosa si intende per sistema blockchain pubblico e privato, cosa è un ledger, cosa si intende per consenso e come viene utilizzata la crittografia in questi sistemi, argomenti contenuti nelle rispettive sottosezioni.

Nella quinta sezione viene descritto in breve cosa si intende per smart contract.

Infine, nella sesta sezione viene descritta l'interoperabilità tra blockchain e vengono trattati i diversi pattern e proprietà esistenti dell'interoperabilità, nelle sottosezioni dedicate.

3.1 PERCHÉ USARE BLOCKCHAIN

Il problema del roaming cellulare può essere gestito con un approccio blockchain [4]. Le procedure di roaming possono essere descritte come diversi smart contract tra l'utente in roaming, HNO e VNO. Inoltre, il meccanismo di consenso nella

blockchain gioca un ruolo indispensabile risolvendo il problema della fiducia. Ciò si ottiene identificando le parti autorizzate a inserire il blocco successivo nella blockchain. Le informazioni registrate su una blockchain possono essere solo inserite, grazie a tecniche crittografiche che garantiscono che, una volta che le transazioni sono state aggiunte al ledger, non possano essere modificate, questa è chiamata proprietà di immutabilità. La tecnologia blockchain può essere utilizzata per automatizzare e monitorare determinate transazioni di stato attraverso l'uso di smart contract senza l'ausilio di terze parti, garantendo una migliore scalabilità. In più, HNO potrebbe usare un sistema basato su tecnologia blockchain di un determinato tipo, mentre VNO potrebbe utilizzare un sistema sempre basato su tecnologia blockchain ma di tipo differente, in tal caso viene garantita la comunicazione tra i due sistemi eterogenei tramite interoperabilità tra blockchain diverse.

Una blockchain privata è una rete di nodi autorizzati a partecipare alla rete. In caso di roaming, sarebbe una rete di nodi blockchain di proprietà di operatori che fanno parte della rete, e ogni nodo blockchain avrà bisogno dell'autorizzazione per visualizzare una determinata transazione che si trova su un nodo. Quindi, una transazione tra due abbonati in roaming può essere vista solo da coloro autorizzati. Essendo una rete blockchain, fornisce ancora la capacità agli operatori di trattare con molti partner di roaming, senza dover costruire un'infrastruttura personalizzata per gestire il roaming con ciascun partner. A causa della mancanza di fiducia tra i partner di roaming, vengono utilizzati degli intermediari nel roaming attuale. In una soluzione blockchain, ci sarà un database distribuito per ogni partner di roaming che registra ogni transazione in modo immutabile. Entrambe le parti possono vedere la transazione e non possono modificarle una volta inserite nella rete blockchain. Ciò aumenta la trasparenza tra i partner di roaming, migliorando la fiducia complessiva. Il ledger distribuito può registrare dati sia transazionali che finanziari

tra le due parti. Questo può essere utilizzato anche per regolare le transazioni finanziarie tra i partner di roaming. In questo modo, la tecnologia blockchain renderà più facile per i partner in roaming lavorare direttamente tra loro senza la necessità di intermediari e infrastrutture derivanti.

I nodi blockchain tra le parti in roaming vengono sincronizzati immediatamente, quindi, tutte le transazioni inserite da un operatore vengono sincronizzate con il database del suo partner di roaming in tempo reale. Ciò rende più facile per gli operatori sapere cosa sta succedendo e monitorare le attività di roaming dell'abbonato.

La sicurezza offerta dalla blockchain insieme allo smart contract può essere utilizzata per ridurre le frodi in roaming. Se un utente ruba una scheda SIM da un dispositivo per effettuare chiamate telefoniche e trasmettere dati in uno scenario di roaming. A questo punto la rete domestica invia uno smart contract alla rete ospitante che la informa che questo dispositivo dovrebbe ricevere solo piccole quantità di dati. La rete ospitante si rende conto che i dati utilizzati non sono conformi alle norme per questo dispositivo e smette di servire le sue richieste. Inoltre, la rete ospitante sarà in grado di contrassegnare questo dispositivo per frode, che avvisa immediatamente anche la rete domestica. Ciò riduce la potenziale frode in tempo reale e riduce al minimo l'impatto finanziario di tali attività.

3.2 COSA È UNA BLOCKCHAIN

Una blockchain è essenzialmente una struttura di dati di sola aggiunta gestita da un insieme di nodi che non si fidano completamente l'uno dell'altro. I nodi nella blockchain concordano su un insieme ordinato di blocchi, ciascuno contenente più transazioni, quindi la blockchain può essere vista come un registro di transazioni

ordinate. Nel contesto del database, la blockchain può essere vista come una soluzione per la gestione delle transazioni distribuite: i nodi conservano le repliche dei dati e concordano un ordine di esecuzione delle transazioni [6].

La blockchain è anche conosciuta come Distributed Ledger Technology (DLT), nel senso che permette di gestire l'aggiornamento dei dati con la collaborazione dei partecipanti alla rete e con la possibilità di avere dati condivisi, accessibili e distribuiti a tutti i partecipanti (o peer). Quest'ultimi hanno la stessa copia identica di questo database. Ciò è in contrasto con le tradizionali architetture centralizzate, in cui, ad esempio, esiste un'unica autorità responsabile di tutte le transazioni in arrivo [5].

Nel design originale, la blockchain di Bitcoin memorizza le monete (o coin) come afferma il sistema. Per questa applicazione, i nodi Bitcoin implementano un semplice modello di macchina a stati replicato che sposta le monete da un indirizzo a un altro. Da allora, la blockchain è cresciuta oltre le criptovalute per supportare stati definiti dall'utente.

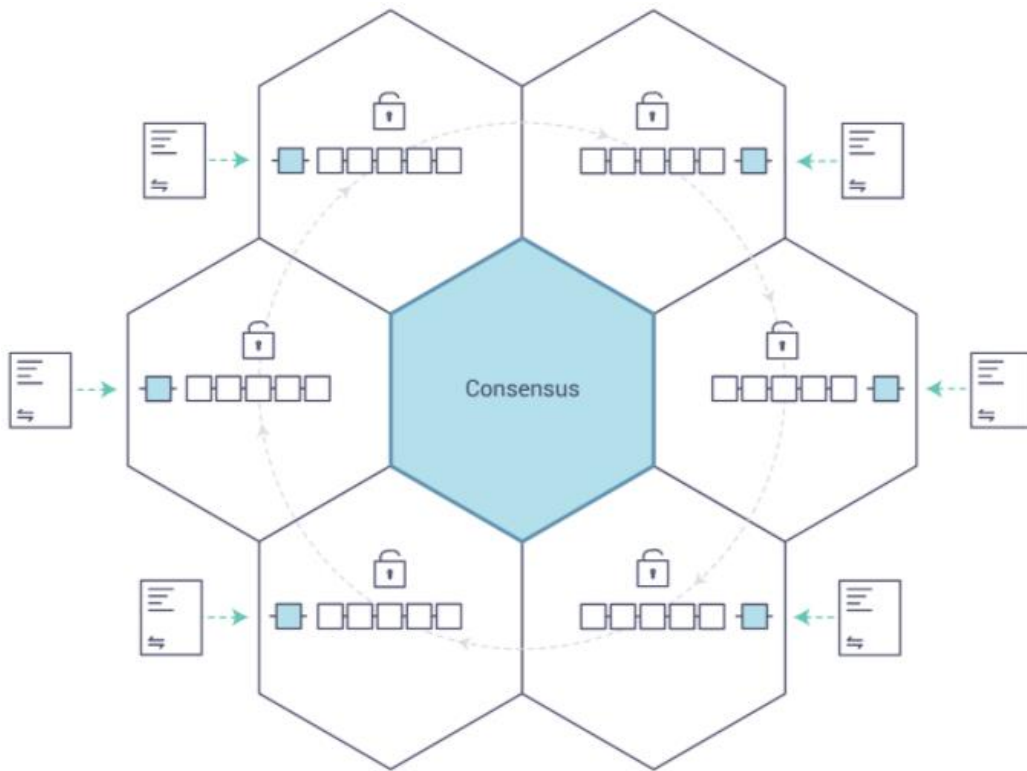


Figura 8. Rappresentazione generica di un sistema blockchain

3.3 CARATTERISTICHE PRINCIPALI DI UNA BLOCKCHAIN

Le caratteristiche principali delle tecnologie blockchain sono l'immutabilità del ledger, la trasparenza, la tracciabilità delle transazioni e la sicurezza basata su funzioni crittografiche [5]. Nessuno può modificare o cancellare la cronologia delle transazioni, fornendo immutabilità e trasparenza.

Gli utenti di una blockchain possono facilmente tenere traccia della cronologia di qualsiasi transazione, poiché tutte le transazioni di una blockchain sono “firmate” digitalmente. Ogni volta che si verifica una transazione, entra in rete, gli algoritmi ne determinano l'autenticità e se è ritenuta valida. Questa nuova transazione è legata

alla precedente formando una catena di transazioni, appunto una blockchain. L'uso della blockchain ha diversi vantaggi. Chiaramente, uno dei maggiori vantaggi è la natura distribuita della blockchain, perché è quasi impossibile manomettere i dati rispetto al database convenzionale. Altri vantaggi forniti dalla natura distribuita sono che è in grado di resistere a qualsiasi attacco alla sicurezza e anche che gli utenti possono essere sicuri che una transazione verrà eseguita come da protocollo. È quasi impossibile violare una rete, perché l'attaccante dovrebbe avere un impatto sulla rete se e solo se ottiene il controllo su quasi il 51% dei nodi, questo è l'unico modo in cui può ingannare il meccanismo di consenso. Ciò significa che la blockchain è resistente agli attacchi informatici, perché funzionerebbe anche quando alcuni nodi sono offline o sotto attacco.

In particolare, l'immutabilità e la trasparenza della blockchain aiutano a ridurre gli errori umani e la necessità di un intervento manuale a causa di dati contrastanti. Blockchain può aiutare a snellire i processi aziendali rimuovendo gli sforzi nella governance dei dati.

3.4 STRUTTURA DI UNA BLOCKCHAIN

Un tipico sistema blockchain è costituito da più nodi che mantengono un insieme di stati globali condivisi ed eseguono transazioni modificando gli stati stessi [6]. Blockchain è una struttura dati speciale che memorizza: stati e cronologia delle transazioni. Tutti i nodi del sistema concordano sulle transazioni e sul loro ordine.

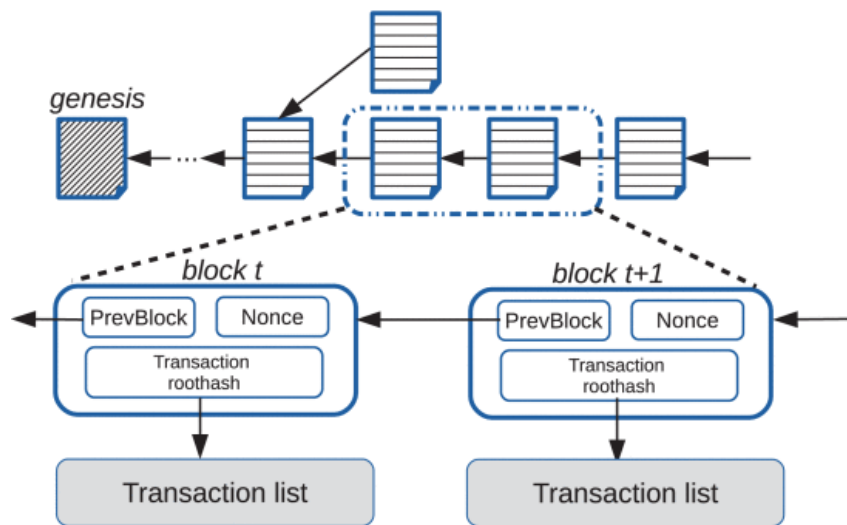


Figura 9. Struttura di una blockchain

Ogni blocco di dati della blockchain è collegato al suo predecessore tramite un puntatore crittografico, fino al primo blocco (chiamato blocco genesis). Per questo motivo, la blockchain viene spesso definita distributed ledger.

3.4.1 BLOCKCHAIN PUBBLICA (O SENZA AUTORIZZAZIONE)

Un sistema blockchain pubblico (o senza autorizzazione), qualsiasi nodo può entrare e uscire dal sistema, quindi la blockchain è completamente decentralizzata, simile a un sistema peer-to-peer [6]. Bitcoin è l'esempio più noto di blockchain pubbliche, dove gli stati sono monete digitali (criptovalute) e una transazione sposta monete da un insieme di indirizzi a un altro. Ogni nodo trasmette una serie di transazioni che desidera eseguire. Nodi speciali chiamati “miners” raccolgono le transazioni in blocchi, ne verificano la validità e avviano un protocollo di consenso per aggiungere i blocchi alla blockchain. Bitcoin utilizza la proof-of-work (PoW) per il consenso: solo un “miner” che ha risolto con successo un puzzle difficile dal

punto di vista computazionale può aggiungere alla blockchain. La PoW è di natura probabilistica, ovvero è possibile che due blocchi vengano aggiunti contemporaneamente, creando un fork nella blockchain. Bitcoin risolve questo problema solo considerando un blocco come confermato dopo che è seguito da un numero di blocchi (in genere sei blocchi). Questa garanzia probabilistica porta a problemi di sicurezza e prestazioni: gli attacchi sono stati dimostrati da un avversario che controlla solo il 25% dei nodi e il throughput delle transazioni Bitcoin rimane molto basso.

La maggior parte dei sistemi blockchain pubblici utilizza varianti di PoW per il consenso, il che funziona bene negli ambienti pubblici perché protegge da specifici attacchi. Tuttavia, essendo non deterministico e computazionalmente costoso, non è adatto per applicazioni come banche e finanza che devono gestire grandi volumi di transazioni in modo deterministico.

3.4.2 BLOCKCHAIN PRIVATA (O CON AUTORIZZAZIONE)

Un sistema blockchain privato (o con autorizzazione) impone una stretta adesione [6]. Più specificamente, esiste un meccanismo di controllo degli accessi per determinare chi può unirsi al sistema. Di conseguenza, ogni nodo viene autenticato e la sua identità è nota agli altri nodi. Hyperledger è tra le blockchain private più popolari. Poiché le identità dei nodi sono note nelle impostazioni private, la maggior parte delle blockchain adotta uno dei protocolli della vasta letteratura sul consenso distribuito.

Hyperledger utilizza, come protocollo di consenso, il PBFT (Protocol Byzantine fault tolerant) che è un protocollo in tre fasi. Nella fase di pre-preparazione, un

leader trasmette un valore da impegnare nel ledger degli altri nodi. Successivamente, nella fase di preparazione, i nodi trasmettono i valori che stanno per eseguire il commit. Infine, la fase di commit conferma tale valore quando più di due terzi dei nodi concordano nella fase precedente. Oltre al consenso deterministico, un'altra proprietà chiave delle blockchain private è che supportano gli smart contract che possono esprimere logiche di transazione altamente complesse. Queste proprietà sono particolarmente desiderabili nei sistemi aziendali e finanziari.

3.4.3 DISTRIBUTED LEDGER

Il ledger è una struttura dati costituita da un elenco ordinato di transazioni, ad esempio, può registrare transazioni monetarie tra più banche o merci scambiate tra parti note [6]. Nelle blockchain, il ledger viene replicato su tutti i nodi. Inoltre, le transazioni sono raggruppate in blocchi che vengono poi concatenati insieme. Pertanto, il ledger è essenzialmente una struttura di dati di sola aggiunta replicata. Una blockchain inizia con alcuni stati iniziali e il ledger registra l'intera cronologia delle operazioni di aggiornamento che vengono effettuate. L'applicazione che utilizza il ledger determina il modello di dati di ciò che viene memorizzato in esso. Il modello di dati acquisisce le astrazioni dei dati chiave, rendendo più semplice per l'applicazione esprimere la propria logica. Ad esempio, un'applicazione di criptovaluta può adottare il modello di conto utente simile ai sistemi bancari tradizionali. D'altra parte, una blockchain generica può utilizzare un modello di basso livello come tabella o coppie chiave-valore. In secondo luogo, il sistema può avere uno o più ledger che possono essere collegati tra loro. Una grande impresa, ad esempio, può possedere più ledger, uno per ciascuno dei suoi dipartimenti:

ingegneria, assistenza clienti, ecc. In terzo luogo, la proprietà del ledger può variare da completamente aperta al pubblico a strettamente controllata da un insieme specifico di nodi. Bitcoin, ad esempio, è completamente aperto, e di conseguenza richiede un protocollo di consenso costoso per identificare chi può aggiornare il ledger. Hyperledger Fabric, d'altra parte, predetermina un insieme di proprietari che possono scrivere sul ledger.

3.4.4 CONSENSO

Il contenuto del ledger contiene la cronologia delle transazioni e i valori attuali mantenuti dalla blockchain. Essendo replicati, gli aggiornamenti effettuati nel ledger devono essere concordati da tutte le parti. In altre parole, queste parti devono raggiungere un consenso [6].

Una proprietà chiave di un sistema blockchain è che i nodi non si fidano l'uno dell'altro, il che significa che alcuni potrebbero comportarsi in modi “byzantine”. Il protocollo di consenso deve quindi tollerare i “byzantine failure”. La letteratura di ricerca sul consenso distribuito è vasta e ci sono molte varianti di questi protocolli. Una parte di questi protocolli è costituito da protocolli puramente basati su calcoli che utilizzano la Proof-of-Work per selezionare casualmente un nodo che decide da sola l'operazione successiva. Il PoW di Bitcoin è un esempio. L'altro insieme dei protocolli sono i protocolli basati esclusivamente sulla comunicazione in cui i nodi hanno voti uguali e passano attraverso più cicli di comunicazione per raggiungere il consenso. Questi protocolli, i PBFT (Practical Byzantine Fault Tolerance), vengono utilizzati in ambienti privati perché assumono nodi autenticati.

Tra l'altro, esistono protocolli ibridi che mirano a migliorare le prestazioni del PoW e PBFT. Alcuni vengono utilizzati nelle blockchain pubbliche per affrontare l'inefficienza del PoW.

3.4.5 CRITTOGRAFIA

I sistemi blockchain fanno un uso massiccio di tecniche crittografiche per garantire l'integrità dei ledger [6]. L'integrità qui si riferisce alla capacità di rilevare la manomissione dei dati nella blockchain. Questa proprietà è vitale nelle strutture pubbliche in cui non esiste una fiducia prestabilita. Ad esempio, la fiducia degli utenti nelle criptovalute come Bitcoin, che determina i valori delle valute, si basa sull'integrità del ledger. Anche nelle blockchain private, l'integrità è altrettanto essenziale perché i nodi autenticati possono comunque agire in modo dannoso.

Esistono almeno due livelli di protezione dell'integrità. Innanzitutto, gli stati globali sono protetti da un albero hash il cui hash radice è archiviato in un blocco. Qualsiasi cambiamento di stato si traduce in un nuovo hash di root. Le foglie dell'albero contengono gli stati, i nodi interni contengono gli hash dei loro figli.

In secondo luogo, la cronologia dei blocchi è protetta, ovvero i blocchi sono immutabili una volta aggiunti alla blockchain. La tecnica chiave è collegare i blocchi attraverso una catena di puntatori hash crittografici: il contenuto del numero di blocco $n+1$ contiene l'hash del numero di blocco n . In questo modo, qualsiasi modifica al blocco n invalida immediatamente tutti i blocchi successivi.

Tramite queste tecniche, blockchain offre un modello di dati sicuro ed efficiente che tiene traccia di tutte le modifiche apportate agli stati globali.

Il modello di sicurezza della blockchain utilizza la crittografia a chiave pubblica. Le identità, comprese quelle degli utenti e delle transazioni, derivano dai certificati

a chiave pubblica. La gestione sicura delle chiavi, quindi, è essenziale per qualsiasi blockchain. Come in altri sistemi di sicurezza, perdere le chiavi private significa perdere l'accesso.

3.5 SMART CONTRACT

Uno smart contract si riferisce al calcolo effettuato quando viene eseguita una transazione [6]. Gli input, gli output e gli stati interessati dall'esecuzione dello smart contract sono concordati da ogni nodo. Tutti i sistemi blockchain hanno smart contract integrati che implementano le loro logiche di business.

Gli smart contract sono altamente personalizzabili, ad esempio, si può scegliere il linguaggio di programmazione utilizzato, la logica da implementare, ecc.

3.6 INTEROPERABILITÀ TRA BLOCKCHAIN

Nella letteratura odierna il concetto reale di interoperabilità non è ancora ben definito, tuttavia si immagina l'interoperabilità come la definizione di smart contract tra due chain diverse in un singolo distributed ledger o attraverso soluzioni eterogenee [5]. Questo significa che occorre cambiare in modo sicuro lo stato delle due chain coinvolte. Quindi l'interoperabilità tra blockchain implica transazioni di stato sicure attraverso diverse blockchain eseguendo applicazioni cross-chain decentralizzate. Le blockchain possono interoperare se queste possono scambiarsi mutuamente informazioni in un modo unificato. In definitiva, l'interoperabilità non

indica propriamente che dei dati formalismi di smart contract o di linguaggi di programmazione usati in una piattaforma possano essere eseguiti su un'altra.

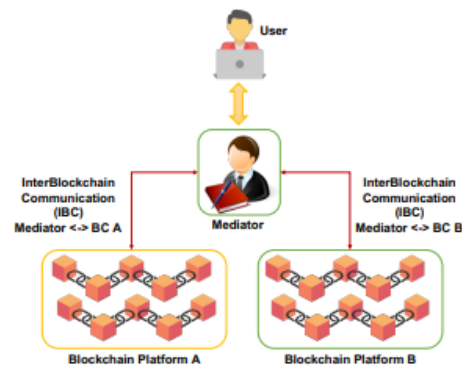
3.6.1 SOLUZIONI DI INTEROPERABILITÀ ESISTENTI

Le principali soluzioni esistenti hanno analizzato la possibilità di trasferire un asset (come una quantità di criptovaluta o determinate risorse) da una piattaforma blockchain ad un'altra. Queste soluzioni possono essere raggruppate in tre tipologie diverse [5]:

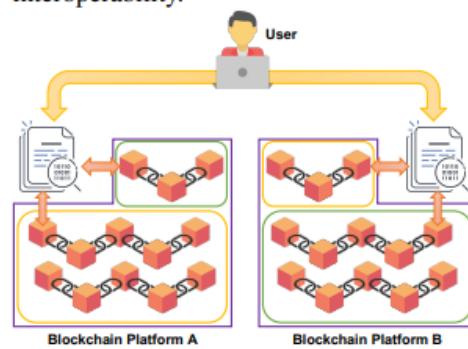
- a) La prima tipologia impiega un approccio centralizzato avendo dei Notary schemes, dove una componente centrale fidata funge da mediator nell'interazione tra le piattaforme, e controlla che l'asset ricevuto da una blockchain venga aggiunto correttamente allo stato dell'altra blockchain. Un tale design è semplice e verifica un trasferimento di asset consistente di fronte a possibili failure ed attacchi. Tuttavia, un tale approccio centralizzato si oppone al fattore chiave che è dietro tutte le piattaforme di blockchain decentralizzate.
- b) La seconda tipologia comprende soluzioni che implementando Sidechain o relay schemes, dove un dato smart contract sul primo ledger può accedere ad asset o ad uno stato mantenuto dalla seconda blockchain ed eseguire computazioni basate su questi. Ciò è possibile in quanto viene mantenuto un sunto dello stato della prima blockchain sulla seconda blockchain, e viceversa: tale implementazione è chiamata Sidechain. Un tale design è completamente decentralizzato e non richiede un mediator centrale fidato. È possibile avere una interoperabilità unidirezionale o bidirezionale, in base al fatto che una blockchain contenga una Sidechain dell'altra blockchain o meno.

- c) La terza tipologia consiste in schemi di hash-locking per un trasferimento di asset decentralizzato e consistente tra blockchain, dove vengono avviati simultaneamente dei cambiamenti di stato atomici su entrambe le blockchain, ed ambedue i cambiamenti vengono finalizzati solo se ciò viene fatto su entrambe, altrimenti si esegue un rollback.

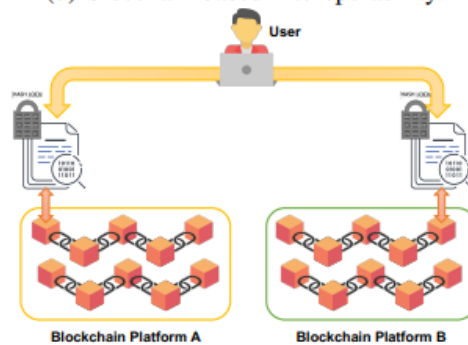
3. BLOCKCHAIN-BASED ROAMING



(a) Notary scheme mediating blockchain interoperability.



(b) Sidechain-based interoperability.



(c) Hash-locking smart contracts for asset transfer between blockchains.

Figura 10. Tipologie di soluzione di interoperabilità tra blockchain

La maggior parte delle soluzioni disponibili appartengono alla seconda tipologia, in quanto sono le più semplici da implementare senza dover rompere la natura decentralizzata delle blockchain. Tuttavia, esse richiedono un cambiamento alla

blockchain interconnessa, così come devono essere introdotti dei meccanismi di locking.

3.6.2 PROPRIETÀ DELLE SOLUZIONI DI INTEROPERABILITÀ

Idealmente, le soluzioni di interoperabilità dovrebbero fornire diverse proprietà, al fine di garantire che le proprietà della blockchain non vengano alterate. Tali proprietà sono [5]:

1. Dipendenze di terze parti:

Le blockchain sono di natura decentralizzata e questo significa che non ci sono dipendenze da terze parti. Anche la soluzione di interoperabilità non dovrebbe fornire alcuna dipendenza da terze parti, al fine di evitare il single point of failure e sovraccaricare la singola entità.

2. Scalabilità:

Questa proprietà si riferisce al numero di transazioni che possono essere elaborate al secondo. Se la soluzione di interoperabilità non sarà scalabile, la comunicazione tra blockchain interoperabili potrà essere influenzata negativamente dall'aumento delle transazioni in arrivo.

3. Sicurezza:

La soluzione di interoperabilità dovrebbe essere estremamente sicura, come la blockchain stessa. Se la soluzione non sarà sicura, di conseguenza la comunicazione non sarà protetta e tutti potrebbero generare problemi, inviando pacchetti manipolati.

CAPITOLO 4

4. HYPERLEDGER FABRIC

In questo capitolo viene introdotto e approfondito la tecnologia principalmente utilizzata in questa tesi, ovvero Hyperledger Fabric.

Nella prima sezione vengono elencate le caratteristiche principali che differenziano Hyperledger Fabric dalle altre DLT, dopo di che, nelle sottosezioni dedicate, viene illustrato il nuovo approccio adottato da Fabric e il suo punto di forza che tratta la privacy e la riservatezza dei dati.

Nella seconda sezione viene illustrata l'architettura di una rete blockchain di tipi Hyperledger Fabric.

Nella terza sezione viene spiegato come vengono riconosciuti gli attori all'interno di una rete Fabric e come funziona tale riconoscimento, approfondite nelle sottosezioni.

Nella quarta sezione viene spiegato come vengono fornite le credenziali agli attori affinché partecipino ad una rete Hyperledger Fabric.

Nella quinta sezione viene spiegato come viene fornito l'accesso ad una data risorsa in una rete Fabric.

Nella sesta sezione vengono illustrati gli elementi fondamentali di una rete Fabric.

Nella settima sezione viene illustrata la struttura di un ledger all'interno di una rete Fabric, spiegando come è strutturato nelle sottosezioni dedicate.

Nell'ottava sezione viene illustrato l'insieme di nodi che è responsabile dell'ordinamento delle transazioni e della loro distribuzione, successivamente viene spiegato come avviene il processo di aggiornamento del ledger nella rispettiva sottosezione.

Infine, nella nona sezione viene illustrato come viene definita la logica di business in un sistema blockchain Fabric.

4.1 COSA È HYPERLEDGER FABRIC

Hyperledger Fabric è una piattaforma DLT (Distributed Ledger Technology) autorizzata di livello aziendale open source, progettata per l'uso di contesti aziendali, che offre alcune funzionalità chiave di differenziazione rispetto ad altre popolari piattaforme di registro distribuito o blockchain [8].

Una prima caratteristica che lo differenzia dalle altre DLT è che Hyperledger è stato fondato sotto la Linux Foundation, che a sua volta ha una storia di grande successo nel coltivare progetti open source. Hyperledger è governato da un comitato direttivo tecnico diversificato, mentre il progetto Hyperledger Fabric da una serie diversificata di manutentori derivanti da più organizzazioni.

Una seconda caratteristica è che Fabric ha un'architettura altamente modulare e configurabile, che consente innovazione, versatilità e ottimizzazione per un'ampia gamma di casi d'uso del settore, tra cui banche, finanza, assicurazioni, sanità, risorse umane, supply chain e persino la consegna di musica digitale. Ad un livello elevato, Fabric è composto dai seguenti componenti modulari:

- Un ordering service, che stabilisce il consenso sull'ordine delle transazioni e quindi trasmette i blocchi ai peer;
- Un membership service provider, che è responsabile dell'associazione di entità nella rete con identità crittografiche;
- Un peer-to-peer gossip service opzionale, che diffonde l'output dei blocchi dell'ordering service ad altri peer;

- Gli smart contract (o “chaincode”), che vengono eseguiti all'interno di un ambiente container (ad esempio Docker). Possono essere scritti in linguaggi di programmazione standard ma non hanno accesso diretto allo stato del ledger;
- Un ledger, che può essere configurato per supportare una varietà di DBMS;
- Una politica di validazione e approvazione, che può essere configurata indipendentemente dall'applicazione.

Una terza caratteristica è che Fabric è la prima piattaforma di ledger distribuito a supportare smart contract creati in linguaggi di programmazione generici come Java, Go e Node.js, piuttosto che in linguaggi specifici del dominio, chiamati Domain-Specific Language (DSL), vincolati. Ciò significa che la maggior parte delle aziende ha già le competenze necessarie per sviluppare smart contract e non è necessaria alcuna formazione aggiuntiva per imparare un nuovo linguaggio di programmazione o DSL.

Una quarta caratteristica è che la piattaforma Fabric è autorizzata (permissioned), il che significa che i partecipanti sono noti tra loro, identificati e spesso controllati. Una blockchain autorizzata fornisce un modo per proteggere le interazioni tra un gruppo di entità che hanno un obiettivo comune ma che potrebbero non fidarsi completamente l'una dell'altra. Facendo affidamento sulle identità dei partecipanti, una blockchain autorizzata può utilizzare i protocolli di consenso più tradizionali, come il Crash Fault Tolerant (CFT) o Byzantine Fault Tolerant (BFT) che non richiedono un mining costoso. Inoltre, in un contesto così autorizzato, il rischio che un partecipante introduca intenzionalmente codice dannoso tramite uno smart contract è ridotto. In primo luogo, i partecipanti sono noti gli uni agli altri con tutte le loro azioni, che si tratti di inviare transazioni tramite applicazione, modificare la configurazione della rete o distribuire uno smart contract, vengono registrate sulla blockchain seguendo una politica di approvazione stabilita per la rete e il tipo di transazione pertinente. Piuttosto che essere completamente anonimo, il colpevole

può essere facilmente identificato e l'incidente viene gestito secondo i termini del modello di governance. Invece, in una blockchain senza autorizzazione (o permissionless), praticamente chiunque può partecipare e ogni partecipante è anonimo. In un tale contesto, non ci può essere fiducia. Al fine di mitigare questa assenza di fiducia, una blockchain senza autorizzazione impiega tipicamente una criptovaluta nativa "estratta" o commissioni di transazione per fornire incentivi economici per compensare i costi straordinari della partecipazione a una forma di consenso del tipo Byzantine Fault Tolerant basata sulla "Proof of Work" (PoW).

Una quinta caratteristica, nonché molto importante, è il suo supporto per i protocolli di consenso che consentono alla piattaforma di essere personalizzata in modo più efficace per adattarsi a casi d'uso e modelli di fiducia particolari. Fabric può sfruttare protocolli di consenso che non richiedono una criptovaluta nativa per incentivare il mining costoso o per alimentare l'esecuzione di smart contract. La prevenzione di una criptovaluta riduce alcuni significativi vettori di rischio, o attacchi, e l'assenza di operazioni di mining crittografico significa che la piattaforma può essere implementata con all'incirca lo stesso costo operativo di qualsiasi altro sistema distribuito.

La combinazione di queste caratteristiche di progettazione differenzianti rende Fabric una delle piattaforme più performanti oggi disponibili sia in termini di elaborazione delle transazioni che di latenza di conferma delle transazioni, e consente la privacy e la riservatezza delle transazioni e degli smart contract (ciò che Fabric chiama "chaincode") che implementano.

4.1.1 UN NUOVO APPROCCIO

Uno smart contract, o ciò che Fabric chiama "chaincode", funziona come un'applicazione distribuita affidabile che ottiene la sua sicurezza / fiducia dalla blockchain e dal consenso esistente tra i peer, in pratica, uno smart contract è la logica di business di un'applicazione blockchain [8].

In una piattaforma blockchain molti smart contract vengono eseguiti contemporaneamente nella rete e possono essere distribuiti dinamicamente (in molti casi da chiunque), in più, il codice dell'applicazione deve essere considerato non attendibile, potenzialmente persino dannoso.

La maggior parte delle piattaforme blockchain capaci di smart contract esistenti seguono l'architettura order-execute in cui il protocollo di consenso convalida e ordina le transazioni, quindi le propaga a tutti i nodi peer, dopodiché ogni peer esegue le transazioni in sequenza. Gli smart contract, eseguiti in una blockchain che opera con una simile architettura, devono essere deterministici, in caso contrario, il consenso potrebbe non essere mai raggiunto. Per affrontare il problema del non determinismo, molte piattaforme richiedono che gli smart contract siano scritti in un linguaggio non standard o specifico del dominio (come Solidity) in modo che le operazioni non deterministiche possano essere eliminate. Ciò può essere un ostacolo, in quanto richiede che sviluppatori scrivano smart contract in un nuovo linguaggio e questo può portare a errori di programmazione. Inoltre, poiché tutte le transazioni vengono eseguite in sequenza da tutti i nodi, le prestazioni e la scalabilità sono limitate. Il fatto che il codice dello smart contract venga eseguito su ogni nodo del sistema richiede l'adozione di misure complesse per proteggere il sistema da contratti potenzialmente dannosi.

Fabric introduce una nuova architettura per le transazioni che chiama execute-order-validate. Affronta le sfide di flessibilità, scalabilità, prestazioni e riservatezza

affrontate dal modello di order-execute separando il flusso delle transazioni in tre fasi:

1. Execute: eseguire una transazione e verifica la correttezza, in caso positivo la approva;
2. Order: ordina le transazioni tramite un protocollo di consenso scelto;
3. Validate: convalida le transazioni rispetto a una politica di approvazione specifica dell'applicazione prima di inserirle nel ledger.

Questo design si discosta radicalmente dal primo paradigma in quanto Fabric esegue le transazioni prima di raggiungere l'accordo finale sul loro ordine di esecuzione.

In Fabric, una politica di approvazione specifica anche quali nodi peer, o quanti di essi, devono garantire la corretta esecuzione di un determinato smart contract. Pertanto, ogni transazione deve essere eseguita (e approvata) solo dal sottoinsieme dei nodi peer necessari per soddisfare la politica di approvazione della transazione. Ciò consente l'esecuzione parallela aumentando le prestazioni complessive e la scalabilità del sistema. Questa prima fase elimina anche qualsiasi non determinismo, poiché i risultati incoerenti possono essere filtrati prima di ordinare le transazioni.

4.1.2 PRIVACY E RISERVATEZZA

In una rete blockchain pubblica e priva di autorizzazioni che sfrutta PoW per il suo modello di consenso, le transazioni vengono eseguite su ogni nodo. Ciò significa che né può esserci riservatezza del codice degli smart contract, né dei dati delle transazioni che elaborano. Ogni transazione e il codice che la implementa sono

visibili a ogni nodo della rete. Questa mancanza di riservatezza può essere problematica per molti casi d'uso aziendali. Al fine di affrontare questa mancanza, si utilizza la crittografia dei dati che è un approccio per garantire la riservatezza. Tuttavia, in una rete senza autorizzazione che sfrutta PoW per il suo consenso, i dati crittografati si trovano su ogni nodo. Dato abbastanza tempo e risorse di calcolo, la crittografia potrebbe essere interrotta. Per molti casi d'uso aziendali, il rischio che le loro informazioni possano essere compromesse è inaccettabile. In un contesto autorizzato che può sfruttare forme alternative di consenso, si potrebbero esplorare approcci che limitano la distribuzione di informazioni riservate esclusivamente ai nodi autorizzati.

Hyperledger Fabric, essendo una piattaforma autorizzata, consente la riservatezza attraverso la sua architettura utilizzando i “canali”. Nei canali, i partecipanti su una rete Fabric stabiliscono una sottorete in cui ogni membro ha visibilità su un particolare insieme di transazioni. Pertanto, solo i nodi che partecipano a un canale hanno accesso allo smart contract (chaincode) e ai dati trattati, preservando la privacy e la riservatezza di entrambi.

4.2 RETE BLOCKCHAIN IN FABRIC

Innanzitutto, una rete blockchain è un'infrastruttura che fornisce servizi di ledger e smart contract (chaincode) alle applicazioni [9]. Gli smart contract vengono utilizzati per generare transazioni che vengono successivamente distribuite a ogni nodo peer nella rete dove sono immutabilmente registrate sulla loro copia del

ledger. Gli utenti delle applicazioni potrebbero essere utenti finali che utilizzano applicazioni client o amministratori di rete.

Nella maggior parte dei casi, più organizzazioni si uniscono per formare un canale su cui vengono richiamate le transazioni sui chaincode e dove le autorizzazioni sono determinate da un insieme di criteri concordati quando il canale viene originariamente configurato. Inoltre, le politiche possono cambiare nel tempo previo accordo delle organizzazioni.

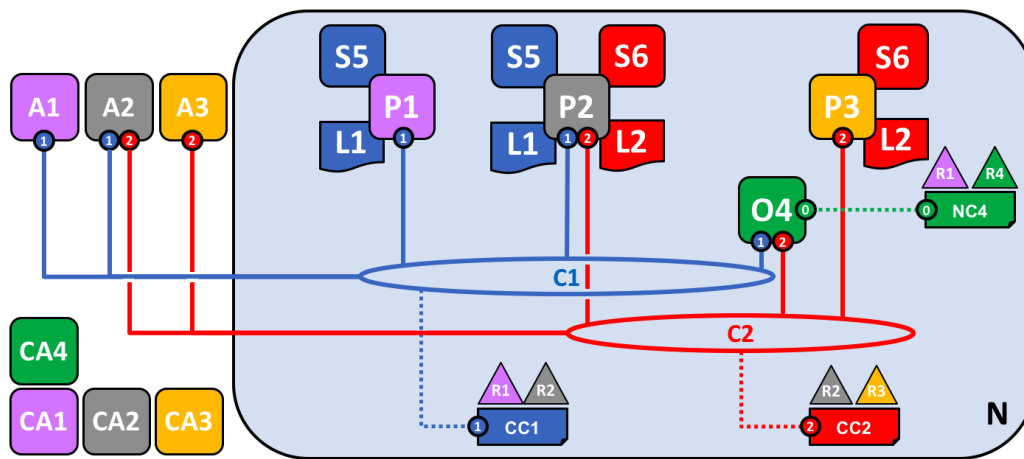


Figura 11. Panoramica dell'architettura di rete di Fabric

La rete N si forma all'avvio di un orderer (o ordering node) che in questo caso è il singolo nodo O4 ed è configurato secondo una configurazione di rete NC4, che conferisce diritti amministrativi all'organizzazione R4. A livello di rete, la Certification Authority CA4 viene utilizzata per distribuire le identità agli amministratori e ai nodi di rete dell'organizzazione R4.

Successivamente, l'organizzazione R4 aggiorna la configurazione di rete per rendere anche l'organizzazione R1 un amministratore. Dopo questo punto R1 e R4 hanno gli stessi diritti sulla configurazione di rete. In più è stata aggiunta la

Certification Authority CA1, che può essere utilizzata per identificare gli utenti dell'organizzazione R1.

Sebbene la rete possa ora essere amministrata da R1 e R4, si può fare ben poco, bisogna creare un consorzio, il quale definisce l'insieme di organizzazioni nella rete che condividono la necessità di effettuare transazioni tra loro. Un amministratore di rete definisce un consorzio X1 che contiene due membri, le organizzazioni R1 e R2. Questa definizione del consorzio è memorizzata nella configurazione di rete NC4 e verrà utilizzata nella fase successiva dello sviluppo della rete.

Ora va creata una parte importante di una rete blockchain Fabric, ovvero il canale, che è un meccanismo di comunicazione primario mediante il quale i membri di un consorzio possono comunicare tra loro. È stato creato un canale C1 per R1 e R2 utilizzando la definizione del consorzio X1. Il canale è governato da una configurazione di canale CC1, completamente separata dalla configurazione di rete. CC1 è gestito da R1 e R2 che hanno gli stessi diritti su C1. R4 non ha alcun diritto su CC1.

In seguito, un nodo peer P1 si è unito al canale C1. I nodi peer sono i componenti di rete in cui sono ospitate le copie del ledger. P1 ospita fisicamente una copia del ledger L1. P1 e O4 possono comunicare tra loro utilizzando il canale C1.

Ora che il canale C1 ha un ledger, possiamo iniziare a connettere le applicazioni client per consumare alcuni dei servizi forniti. Uno smart contract S5 è stato installato su P1. L'applicazione client A1 dell'organizzazione R1 può utilizzare S5 per accedere al ledger tramite il nodo peer P1. A1, P1 e O4 sono tutti uniti al canale C1, cioè possono tutti utilizzare le strutture di comunicazione fornite da quel canale. Seguendo gli stessi criteri possono essere aggiunti successivamente ulteriori peer, applicazioni client, canali, ledger e smart contract.

4.3 IDENTITÀ

Ciascuno attore, ovvero elemento attivo all'interno o all'esterno di una rete in grado di consumare servizi, in una rete blockchain, ha un'identità digitale incapsulata in un certificato digitale [10]. Queste identità sono importanti perché determinano le autorizzazioni esatte sulle risorse e l'accesso alle informazioni che gli attori hanno in una rete blockchain. Affinché un'identità sia verificabile, deve provenire da un'autorità fidata, e in Fabric questa autorità è il Membership Service Provider (MSP).

Più specificamente, un MSP è un componente che definisce le regole che governano le identità valide per una data organizzazione. L'implementazione MSP predefinita in Fabric utilizza i certificati X.509 come identità, adottando un modello gerarchico PKI (Public Key Infrastructure) tradizionale.

4.3.1 PUBLIC KEY INFRASTRUCTURE (PKI)

Un'infrastruttura a chiave pubblica (PKI) è una raccolta di tecnologie Internet che fornisce comunicazioni protette in una rete [10].

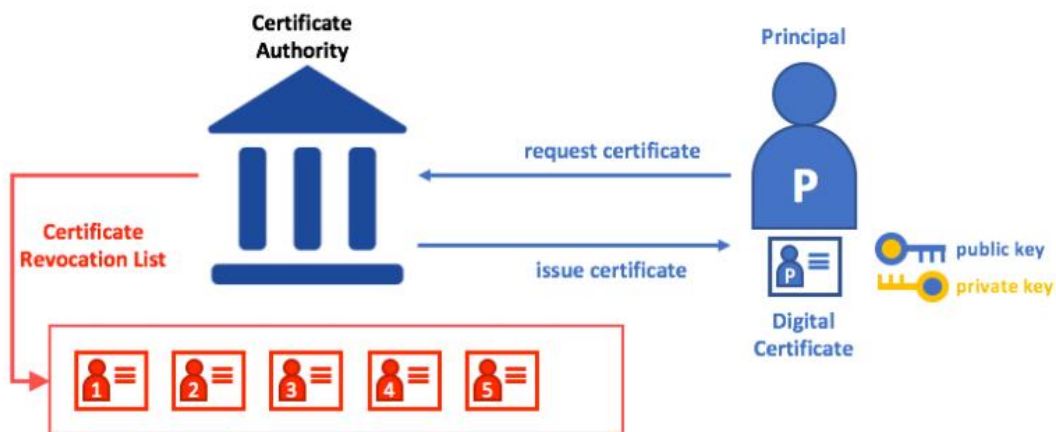


Figura 12. Funzionamento di una PKI

Una PKI è composta principalmente da una o più Certification Authority (CA), che rilasciano certificati digitali che poi vengono utilizzati per autenticare i messaggi scambiati. Un Certificate Revocation List (CRL) di una CA costituisce un riferimento per i certificati non più validi. La revoca di un certificato può avvenire per una serie di motivi. Ad esempio, un certificato può essere revocato perché il materiale privato crittografico associato al certificato è stato esposto.

Sebbene una rete blockchain sia più di una rete di comunicazione, si basa sullo standard PKI per garantire una comunicazione sicura tra i vari partecipanti alla rete e per garantire che i messaggi pubblicati sulla blockchain siano correttamente autenticati.

4.3.2 CERTIFICATI DIGITALI

Un certificato digitale è un documento che contiene una serie di attributi relativi al titolare del certificato [10]. Il tipo di certificato più comune è quello conforme allo standard X.509.

Ciò che è importante è che tutti gli attributi di un certificato possano essere registrati utilizzando una tecnica matematica chiamata crittografia in modo che la manomissione invalidi il certificato intero. La crittografia consente di presentare il certificato ad altri per dimostrare l'identità a condizione che l'altra parte si fidi dell'emittente del certificato, noto come Certification Authority (CA). Finché la CA conserva in modo sicuro determinate informazioni crittografiche (ovvero, la propria chiave privata), chiunque legga il certificato può essere certo che le informazioni al suo interno non siano state manomesse.

4.3.3 CERTIFICATION AUTHORITY (CA)

Le CA sono una parte comune dei protocolli di sicurezza Internet, una Certificate Authority distribuisce i certificati a diversi attori di una rete [10]. Questi certificati sono firmati digitalmente dalla CA e collegano l'attore con una chiave pubblica (e facoltativamente con un elenco completo di proprietà). Di conseguenza, se ci si fida della CA (e si conosce la sua chiave pubblica), ci si può fidare che l'attore specifico è vincolato alla chiave pubblica inclusa nel certificato e possiede gli attributi inclusi, convalidando la firma della CA sul certificato dell'attore.

I certificati possono essere ampiamente diffusi, poiché non includono né le chiavi private degli attori né quelle della CA. In quanto tali, possono essere utilizzati per autenticare i messaggi provenienti da attori diversi. In un ambiente blockchain, ogni attore che desidera interagire con la rete ha bisogno di un'identità.

4.4 MEMBERSHIP SERVICE PROVIDER (MSP)

Poiché Fabric è una rete autorizzata, i partecipanti alla blockchain hanno bisogno di un modo per dimostrare la propria identità al resto della rete per poter effettuare transazioni [11]. Le Certification Authority emettono un'identità generando una chiave pubblica e privata che forma una coppia di chiavi che può essere utilizzata per dimostrare l'identità di un membro della rete. Poiché una chiave privata non può mai essere condivisa pubblicamente, è necessario un meccanismo per la verifica, ed è qui che entra in gioco il Membership Service Provider (MSP). Ad esempio, un peer utilizza la sua chiave privata per firmare digitalmente, o approvare, una transazione. L'MSP sull'ordering service contiene la chiave pubblica del peer che viene quindi utilizzata per verificare che la firma allegata alla transazione sia valida. La chiave privata viene utilizzata per produrre una firma su una transazione che solo la chiave pubblica corrispondente, che fa parte di un MSP, può abbinare. Pertanto, l'MSP è il meccanismo che consente a tale identità di essere considerata attendibile e riconosciuta dal resto della rete senza mai rivelare la chiave privata del membro della rete. Mentre le Certification Authority generano i certificati che rappresentano le identità, l'MSP contiene un elenco di identità autorizzate. Nella pratica, un MSP non fornisce nulla, la sua implementazione è un insieme di cartelle che vengono aggiunte alla configurazione della rete e viene utilizzato per definire un'organizzazione.

4.5 POLITICHE

Al suo livello più elementare, una politica è un insieme di regole che descrivono tipicamente un chi e un cosa, come l'accesso o i diritti che un individuo ha su una risorsa [12]. Le politiche sono una delle cose che rendono Hyperledger Fabric diverso da altri blockchain come Ethereum o Bitcoin. In questi sistemi, le transazioni possono essere generate e convalidate da qualsiasi nodo della rete. Le politiche che governano la rete sono fissate in qualsiasi momento e possono essere modificate solo utilizzando lo stesso processo che governa il codice. Poiché Fabric è una blockchain autorizzata i cui utenti sono riconosciuti dall'infrastruttura sottostante, tali utenti hanno la possibilità di decidere sulla governance della rete prima che venga lanciata e modificare la governance di una rete in esecuzione.

Le politiche consentono ai membri di decidere quali organizzazioni possono accedere o aggiornare una rete Fabric e forniscono il meccanismo per applicare tali decisioni. Queste politiche contengono gli elenchi di organizzazioni che hanno accesso a una determinata risorsa, come un utente o un chaincode di sistema. Specificano anche quante organizzazioni devono concordare una proposta per aggiornare una risorsa, come un canale o smart contract. Una volta scritte, le politiche valutano la raccolta di firme allegate a transazioni e proposte, e convalidano se le firme soddisfano la governance concordata dalla rete.

4.6 I PEER

I peer sono un elemento fondamentale della rete perché ospitano ledger e smart contract [13]. Gli smart contract e i ledger vengono rispettivamente utilizzati per incapsulare processi e informazioni condivise in una rete.

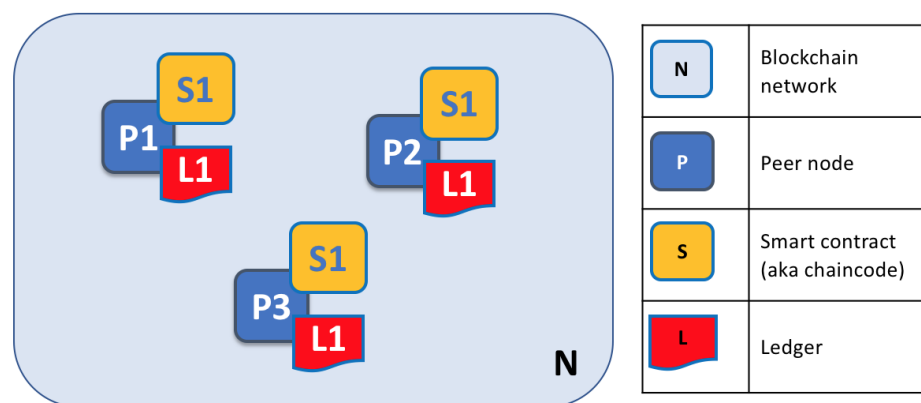


Figura 13. Rappresentazione di peer contenenti smart contract e ledger

I peer possono essere creati, avviati, arrestati, riconfigurati e persino eliminati. Espongono una serie di API che consentono agli amministratori e alle applicazioni di interagire con i servizi che forniscono.

Più nel dettaglio, un peer ospita effettivamente istanze del ledger e istanze di smart contract. In effetti, un peer può anche ospitare più ledger e più smart contract diversi, il che è utile perché consente una progettazione flessibile del sistema.

Le applicazioni si connettono sempre ai peer quando devono accedere al ledger e chaincode. Le API disponibili consentono alle applicazioni di connettersi ai peer, invocare i chaincode per generare transazioni, inviare transazioni alla rete che verranno “ordinate”, convalidate e impegnate nel ledger e ricevere eventi quando

questo processo è completo. Attraverso una connessione peer, le applicazioni possono eseguire chaincode per interrogare o aggiornare un ledger.

Un peer può restituire immediatamente i risultati di una query ad un'applicazione poiché tutte le informazioni richieste per soddisfare la query si trovano nella copia locale del peer. I peer non si consultano mai con altri peer per rispondere a una query da un'applicazione. Le applicazioni possono, tuttavia, connettersi a uno o più peer per inviare una query, ad esempio, per confermare un risultato tra più peer o recuperare un risultato più aggiornato da un altro peer se si sospetta che le informazioni potrebbero non essere aggiornate.

4.7 LEDGER

Un ledger, è un concetto chiave in Hyperledger Fabric, memorizza importanti informazioni sugli oggetti di business, ovvero il valore corrente degli attributi degli oggetti e la cronologia delle transazioni che hanno prodotto tali valori [14]. Infatti, all'interno di un ledger non verrà salvato l'intero oggetto di business ma dei "Facts" sullo stato corrente dell'oggetto e sulla cronologia delle transazioni che hanno portato a tale stato. Questi "Facts" che vengono memorizzati nel ledger, ci consentono di identificare l'oggetto insieme ad altre informazioni chiave su di esso. Sebbene i "Facts" sullo stato attuale di un oggetto possano cambiare, la cronologia su di esso è immutabile, può essere aggiunta ma non modificata.

In Hyperledger Fabric, un ledger è costituito da due parti distinte, sebbene correlate: un world state e una blockchain. Il world state è un database che contiene i valori correnti di un insieme di stati del ledger, questi stati sono espressi come coppie chiave-valore. Invece, una blockchain è un registro di transazioni che annota tutti i cambiamenti che hanno portato all'attuale stato del ledger, più precisamente, le

transazioni vengono raccolte all'interno di blocchi dopo di che aggiunti alla blockchain, ricordando che tali blocchi sono immutabili.

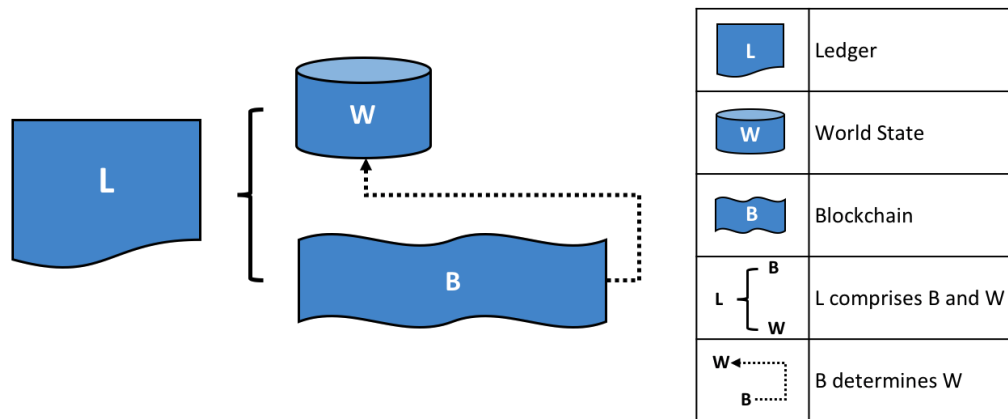


Figura 14. Struttura di un ledger in Fabric

È utile pensare che esista un registro logico in una rete Hyperledger Fabric. In realtà, la rete mantiene più copie di un ledger, che vengono mantenute coerenti con ogni altra copia attraverso un processo chiamato consenso. Il termine Distributed Ledger Technology (DLT) è spesso associato a questo tipo di ledger, logicamente singolare, ma con molte copie coerenti distribuite su una rete.

4.7.1 WORLD STATE

Il world state mantiene i valori correnti degli attributi di un oggetto di business[14]. Ciò è utile perché i programmi di solito richiedono il valore corrente di un oggetto, altrimenti sarebbe complicato attraversare l'intera blockchain per calcolare il valore corrente di tale oggetto.

4. HYPERLEDGER FABRIC

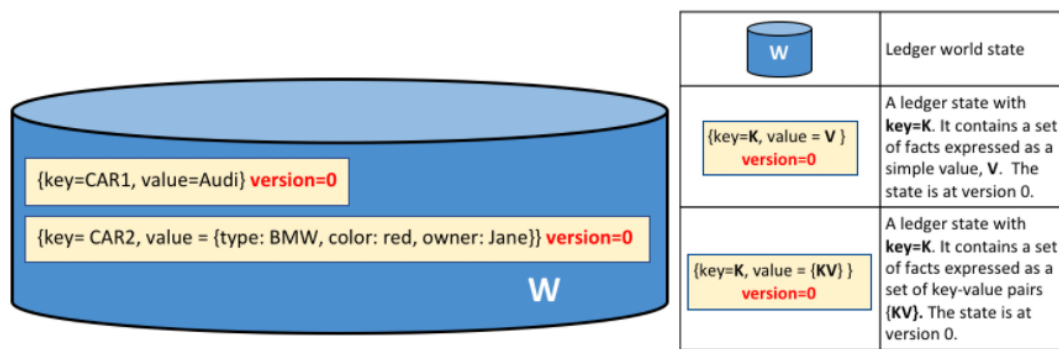


Figura 15. Struttura del world state

Il world state del ledger contenente due stati, il primo è una chiave (key), mentre il secondo è un valore più complesso, il quale può avere più attributi. Riassumendo, uno stato del ledger registra una serie di “Facts” su un particolare oggetto business. Il world state è implementato come un database, il quale fornisce un ricco set di operatori per l'archiviazione e il recupero efficienti degli stati memorizzati. Più in particolare, Hyperledger Fabric può essere configurato per utilizzare diversi database per soddisfare le esigenze di diversi tipi di valori di stato e gli schemi di accesso richiesti dalle applicazioni, ad esempio con query complesse.

Le applicazioni inviano transazioni che effettuano delle modifiche al world state e finiscono per essere impegnate nella blockchain del ledger. Le applicazioni sono isolate dal meccanismo di consenso, queste invocano semplicemente uno smart contract e vengono informati quando la transazione è stata inclusa nella blockchain, ovvero se è stata validata o meno. Il punto chiave della progettazione è che solo le transazioni firmate dall'insieme richiesto di organizzazioni comporteranno un aggiornamento al world state. Se una transazione non è firmata da un numero sufficiente di peer, tale cambiamento non verrà applicato al world state.

Infine, quando un ledger viene creato per la prima volta, il world state è vuoto. Poiché qualsiasi transazione è registrata sulla blockchain, significa che il world state può essere rigenerato in qualsiasi momento. Questo può essere molto

conveniente, ad esempio, viene generato automaticamente quando viene creato un peer. Inoltre, se un peer smette di funzionare in modo anomalo, il world state può essere rigenerato al riavvio del peer, prima che le transazioni vengano accettate.

4.7.2 BLOCKCHAIN (DEL LEDGER)

Mentre il world state contiene una serie di “Facts” relativi allo stato attuale di un insieme di oggetti di business, la blockchain è una cronologia su come questi oggetti sono arrivati al loro stato attuale [14]. La blockchain registra ogni versione precedente di ogni stato del ledger e come è stata modificata.

La blockchain è strutturata come un registro sequenziale di blocchi interconnessi, in cui ogni blocco contiene una sequenza di transazioni, ogni transazione rappresenta una query o un aggiornamento al world state. La sequenza dei blocchi, così come la sequenza delle transazioni all'interno dei blocchi, viene stabilita quando questi vengono creati per la prima volta da un componente Hyperledger Fabric chiamato ordering service.

L'header di ogni blocco include un hash delle transazioni del blocco, nonché un hash dell'intestazione del blocco precedente. In questo modo, tutte le transazioni sul ledger vengono sequenziate e collegate crittograficamente tra loro. Questo hashing rende i dati del ledger molto sicuri. Anche se un nodo che ospita il ledger fosse manomesso, non sarebbe in grado di convincere tutti gli altri nodi che ha la blockchain esatta perché il ledger è distribuito attraverso una rete di nodi indipendenti. La blockchain è implementata come file, a differenza del world state, che utilizza un database.

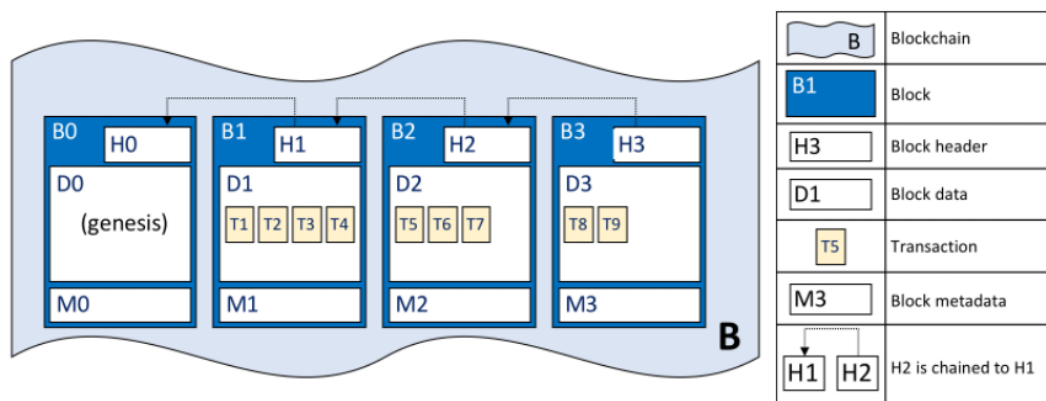


Figura 16. Struttura della blockchain del ledger

Infine, il primo blocco nella blockchain è chiamato blocco genesis, che è il punto di partenza per il ledger, sebbene non contenga alcuna transazione dell'utente. Invece, contiene una transazione di configurazione contenente lo stato iniziale del canale di rete.

4.7.3 BLOCCHI

Un blocco contenuto in una blockchain è composto da [14]:

- **Blocco header:** quando viene creato un nuovo blocco, vengono inseriti nel blocco header delle informazioni, come il numero del blocco creato, che è un numero intero, hash del blocco corrente e hash del blocco precedente. Questi campi sono derivati internamente mediante l'hashing crittografico dei dati del blocco. Assicurano che ogni blocco sia inestricabilmente collegato al suo predecessore, questo assicura che il ledger è immutabile.

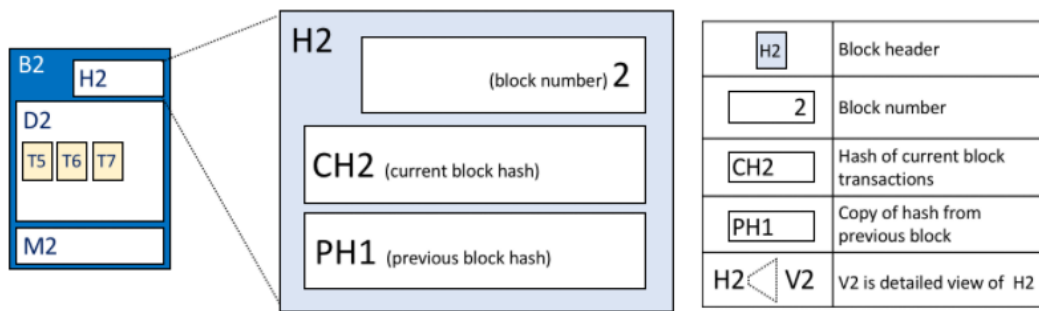


Figura 17. Struttura di un blocco

- Blocco dati: questo blocco contiene un elenco di transazioni disposte in ordine. Viene scritto quando il blocco viene creato dall'ordering service.
- Blocco metadati: questa sezione contiene il certificato e la firma del creatore del blocco che viene utilizzato per verificarlo dai nodi di rete. Successivamente, il block committer aggiunge un indicatore valido / non valido per ogni transazione in una bitmap che risiede anche nei metadati del blocco, nonché un hash degli aggiornamenti di stato cumulativi fino a quel blocco incluso, al fine di rilevare un fork di stato. A differenza dei dati del blocco e dei campi di intestazione, questa sezione non è un input per il calcolo dell'hash del blocco.

4.7.4 TRANSAZIONI

Una transazione contiene i seguenti campi [14]:

- Header: contiene alcuni metadati essenziali sulla transazione, ad esempio il nome del chaincode pertinente e la sua versione;
- Firma: contiene una firma crittografica, creata dall'applicazione client. Questo campo viene utilizzato per verificare che i dettagli della transazione

non siano stati manomessi, in quanto richiede la chiave privata dell'applicazione per generarli.

- **Proposta:** codifica i parametri di input forniti da un'applicazione allo smart contract per creare l'aggiornamento che verrà inserito nel ledger. Quando viene eseguito lo smart contract, questa proposta fornisce una serie di parametri di input che, in combinazione con il world state corrente, determina il nuovo world state.
- **Risposta:** contiene l'output di uno smart contract e, se la transazione viene convalidata con successo, verrà applicata al ledger per aggiornare il world state.
- **Endorsement:** contiene un elenco di risposte firmate da ciascuna organizzazione richiesta sufficiente a soddisfare la politica di approvazione.

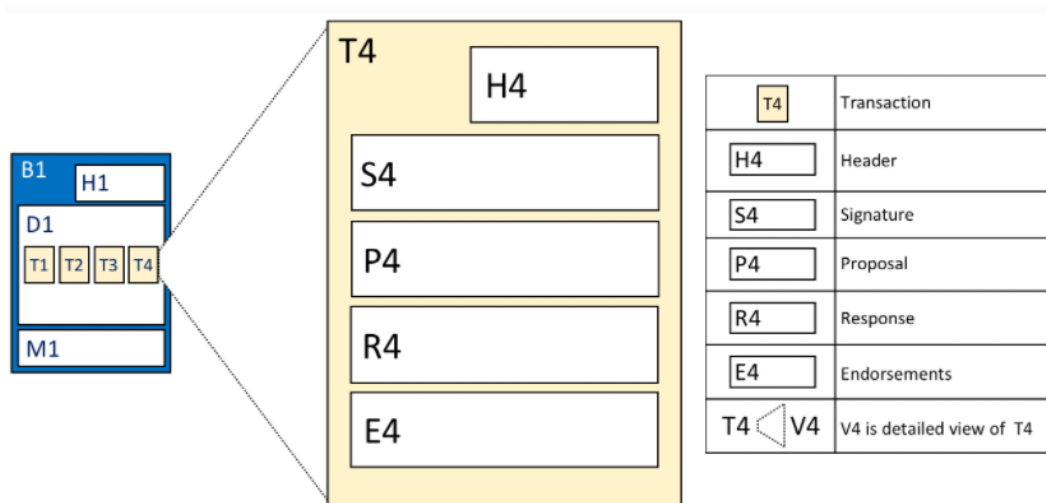


Figura 18. Struttura di una transazione

4.8 ORDERING SERVICE

Molte blockchain distribuite, come Ethereum e Bitcoin, non sono autorizzate, il che significa che qualsiasi nodo può partecipare al processo di consenso, in cui le transazioni vengono ordinate e raggruppate in blocchi [15]. A causa di questo fatto, questi sistemi si basano su algoritmi di consenso probabilistico che alla fine garantiscono la coerenza del ledger con un alto grado di probabilità, ma che sono ancora vulnerabili ai ledger divergenti (noti anche come "fork" del ledger), in cui diversi partecipanti alla rete hanno una visione diversa dell'ordine accettato delle transazioni.

Hyperledger Fabric funziona in modo diverso. È dotato di un nodo chiamato orderer (noto anche come "ordering node") che esegue questo ordinamento delle transazioni, che insieme ad altri nodi di orderer costituisce un ordering service. Poiché il design di Fabric si basa su algoritmi di consenso deterministico, è garantito che qualsiasi blocco convalidato dal peer sia definitivo e corretto. I ledger non possono fare il fork come fanno in molte altre reti blockchain distribuite e senza autorizzazione. Oltre a questo, Fabric offre vantaggi in termini di prestazioni e scalabilità, eliminando i colli di bottiglia che possono verificarsi quando l'esecuzione e l'ordinamento vengono eseguiti dagli stessi nodi.

4.8.1 FLUSSO DELLE TRANSAZIONI

I peer costituiscono la base di una rete blockchain, ospitano ledger che possono essere interrogati e aggiornati dalle applicazioni tramite smart contract, quest'ultimi sempre ospitati dai peer [15]. Nello specifico, le applicazioni che desiderano

aggiornare il ledger sono coinvolte in un processo con tre fasi che garantisce che tutti i peer in una rete blockchain mantengano i loro ledger coerenti tra loro.

- **Fase 1:**

Un'applicazione client invia una proposta di transazione a un sottoinsieme di peer che invocherà uno smart contract per produrre un aggiornamento del ledger e quindi approvare i risultati. I peer che approvano restituiscono una risposta all'applicazione client.

- **Fase 2:**

In questa fase, i client dell'applicazione inviano transazioni contenenti risposte di proposta di transazione approvate a un nodo dell'ordering service, che crea blocchi di transazioni che verranno infine distribuiti a tutti i peer sul canale per la convalida finale e il commit nella fase tre. I nodi dell'ordering service ricevono transazioni da molti client di applicazioni differenti contemporaneamente. Questi nodi lavorano insieme per formare collettivamente l'ordering service. Il suo compito è organizzare batch di transazioni inviate in una sequenza ben definita e impacchettarle in blocchi che verranno aggiunti in seguito alla blockchain.

- **Fase 3:**

Questa fase inizia con l'orderer che distribuisce i blocchi a tutti i peer ad esso collegati. Ogni peer convaliderà i blocchi distribuiti indipendentemente, ma in modo deterministico, assicurando che i ledger rimangano coerenti. In particolare, ogni peer nel canale convaliderà ogni transazione nel blocco per garantire che sia stata approvata dai peer dell'organizzazione richiesta. Le transazioni non convalidate vengono ancora conservate nel blocco immutabile creato dall'orderer, ma sono contrassegnate come non valide dal peer e non aggiornano lo stato del ledger.

4.9 SMART CONTRACT (O CHAINCODE)

Dal punto di vista di uno sviluppatore di applicazioni, uno smart contract , insieme al ledger, costituiscono il cuore di un sistema blockchain di Hyperledger Fabric [16]. Mentre un ledger contiene “Facts” sullo stato attuale e cronologia di un insieme di oggetti di business, uno smart contract definisce la logica eseguibile che genera nuove informazioni che vengono aggiunti al ledger. Un chaincode viene in genere utilizzato dagli amministratori per raggruppare smart contract correlati per la distribuzione, ma può anche essere utilizzato per la programmazione di sistema di basso livello di Fabric. Prima che le aziende possano effettuare transazioni tra loro, devono definire una serie comune di contratti che coprono termini, dati, regole, definizioni di concetti e processi comuni. Nel loro insieme, questi contratti definiscono il modello di business che regola tutte le interazioni tra le parti che effettuano transazioni.

In generale, uno smart contract definisce la logica della transazione che controlla il ciclo di vita di un oggetto di business contenuto nel world state. Viene quindi impacchettato in un chaincode e poi viene distribuito su una rete blockchain.

Le applicazioni invocano uno smart contract per generare transazioni che poi vengono registrate nel ledger.

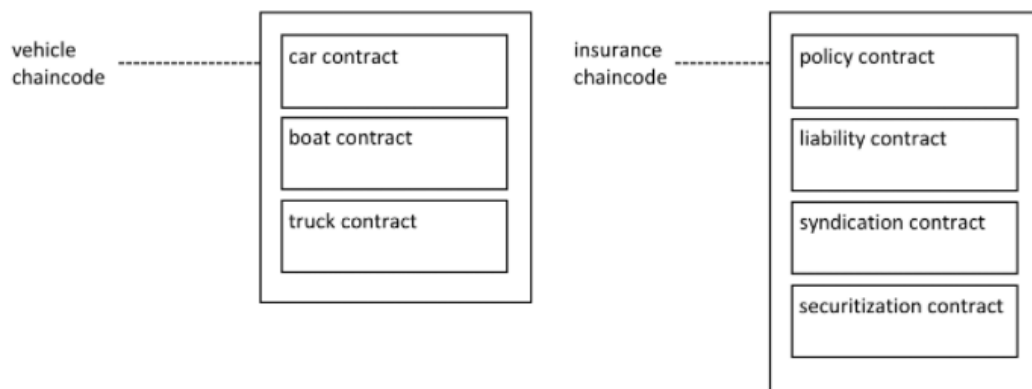


Figura 19. Chaincode contenenti Smart Contract

Più nel dettaglio, uno smart contract è definito all'interno di un chaincode. È possibile definire più smart contract all'interno dello stesso chaincode. Quando viene distribuito un chaincode, tutti gli smart contract al suo interno vengono resi disponibili alle applicazioni.

Uno smart contract accede programmaticamente a due parti distinte del ledger, ovvero alla blockchain, che registra immutabilmente la cronologia di tutte le transazioni, e al world state, che conserva una cache del valore corrente di questi stati, poiché è il valore corrente di un oggetto che è normalmente richiesto. Gli smart contract inseriscono, ottengono ed eliminano principalmente stati nel world state e possono anche interrogare il record, immutabile, delle transazioni ovvero la blockchain.

Ogni smart contract ha una politica di approvazione associata ad esso. Questa politica di approvazione identifica quali organizzazioni devono approvare le transazioni generate dallo smart contract prima che tali transazioni possano essere identificate come valide. Se una politica di approvazione specifica che più di un'organizzazione deve firmare una transazione, lo smart contract deve essere eseguito da un insieme sufficiente di organizzazioni affinché venga generata una transazione valida.

CAPITOLO 5

5. SISTEMA PROPOSTO

In questo capitolo viene spiegato il sistema proposto per risolvere i problemi trattati precedentemente.

Nella prima sezione viene illustrato lo scenario che si intende implementare.

Nella seconda sezione viene spiegato come installare le varie componenti del sistema.

Ed infine, nella terza sezione, viene spiegata l'implementazione del sistema che realizza lo scenario.

5.1 SCENARIO

Questa sezione illustra lo scenario per la tariffazione del servizio di roaming implementato nel progetto di tesi.

L'applicazione realizzata simula lo scenario in cui gli operatori coinvolti facciano parte della stessa rete blockchain, quindi federated operators. Quando un cliente, abbonato ai servizi di un operatore telefonico, non si trova geograficamente nella rete coperta dal suo operatore, supponendo che siano stati siglati degli accordi tra le due reti, per continuare ad utilizzare tali servizi ogni istanza di roaming dev'essere conservata nella blockchain, dove vengono mantenute varie informazioni. Un tale processo è automatizzabile, infatti, si può supporre che ogni istanza venga generata automaticamente al termine di ogni chiamata o utilizzo di dati mobili. Dunque, l'operatore della rete ospitante, al termine del roaming, genera

questa istanza riconoscendo l'operatore della rete domestica, memorizzando numero di cellulare dell'abbonato, data e orario di roaming, durata (in secondi) del roaming e tipologia di servizio (chiamata o dati). Di conseguenza, per convalidare la transazione sono necessarie le firme sia dell'operatore della HPMN, sia dell'operatore della VPMN.

Il processo, essendo automatico, dev'essere astratto dal punto di vista dell'abbonato, che deve essere esclusivamente consapevole di aver utilizzato servizi di roaming. L'applicazione deve soltanto consentire tali istanze nel ledger, astruendo da ulteriori complessi dettagli come, ad esempio, il calcolo del costo dei servizi di roaming, che dipendono da accordi esterni. Questa applicazione è stata progettata per risolvere, attraverso smart contract, le limitazioni individuate nel sistema attuale dovute alla presenza di intermediari quali le clearing house.

5.2 INSTALLAZIONE

Per quanto riguarda il sistema operativo su cui viene eseguito tale progetto, è stato utilizzato Ubuntu 20.04 LTS.

Per la realizzazione dello scenario, ovvero una rete federated operators, è stata utilizzata esclusivamente la tecnologia blockchain Hyperledger Fabric.

Per l'installazione della componente Hyperledger Fabric, è stata seguita la user guide dal sito ufficiale.

5.3 IMPLEMENTAZIONE

In questa sezione viene illustrato e spiegato il codice scritto dell'implementazione dello scenario.

Il sistema che implementa la rete di tipo federated operators, in questa tesi, viene denominato “roaming-data” e viene utilizzata la tecnologia Hyperledger Fabric.

Tale sistema è composto da tre componenti scritti in Javascript, ovvero i due client, che richiedono di inserire transazioni nel ledger, e lo smart contract, che definisce le operazioni che possono essere effettuate.

I client sono i componenti che ascolteranno il socket, in attesa di un messaggio dall'altra estremità del socket. Invece, lo smart contract eseguirà solo le operazioni richieste dai client. Questa sottosezione inizia con la spiegazione del codice dello smart contract, seguita dalla spiegazione del codice dei 2 client.

```

async InitLedger(ctx) {
  var assets = [
    {
      NumTell: '3387032555',
      OwnerRoaming: 'Operatore1',
      DateRoaming: '05/12/2020',
      TimeStartRoaming: '12:36:49',
      Seconds: '3000',
      ServiceType: 'internet',
    },
    {
      NumTell: '3398043666',
      OwnerRoaming: 'Operatore1',
      DateRoaming: '04/11/2020',
      TimeStartRoaming: '21:10:11',
      Seconds: '1000',
      ServiceType: 'call',
    },
  ];
  for (const asset of assets) {
    asset.docType = 'asset';
    var key = producerKey(asset.NumTell, asset.DateRoaming, asset.TimeStartRoaming);
    await ctx.stub.putState(key, Buffer.from(JSON.stringify(asset)));
    console.info(`Asset ${key} initialized`);
  }
}

```

Figura 20. Metodo per inizializzare il ledger

Il metodo illustrato nella *Figura 20*, ovvero `InitLedger`, rappresenta la funzione che popola il ledger con alcuni asset iniziali. Notare che al suo interno viene esplicitata anche la struttura degli asset che il sistema gestirà. Successivamente è presente un ciclo e al suo interno viene utilizzato l'oggetto `stub`. Quest'ultimo incapsula le API tra l'implementazione del chaincode e il peer Fabric, in pratica, è l'oggetto responsabile dell'aggiornamento dello stato della blockchain.

```
async CreateAsset(ctx, numTell, ownerRoaming, dateRoaming, timeStartRoaming, seconds, serviceType) {
  const asset = {
    NumTell: numTell,
    OwnerRoaming: ownerRoaming,
    DateRoaming: dateRoaming,
    TimeStartRoaming: timeStartRoaming,
    Seconds: seconds,
    ServiceType: serviceType
  };
  var key = producerKey(numTell, dateRoaming, timeStartRoaming);
  ctx.stub.putState(key, Buffer.from(JSON.stringify(asset)));
  return JSON.stringify(asset);
}
```

Figura 21. Metodo per create un nuovo asset

Il metodo illustrato nella *Figura 21*, ovvero `CreateAsset`, rappresenta la funzione per creare un nuovo asset all'interno del ledger che non esiste ancora. Successivamente si interagisce con l'oggetto `stub`, così che l'asset appena creato venga inserito all'interno del ledger.

```
async ReadAsset(ctx, numTell) {
  const assetJSON = await ctx.stub.getState(numTell);
  if (!assetJSON || assetJSON.length === 0) {
    throw new Error(`The asset ${numTell} does not exist`);
  }
  return assetJSON.toString();
}
```

Figura 22. Metodo per leggere un asset all'interno del ledger

Ora che sono stati creati i metodi di inserimento degli asset e inizializzazione del ledger, viene illustrato il metodo nella *Figura 22*, ovvero ReadAsset, che consente di leggere un asset presente nel ledger. Prima di tutto, viene verificato che tale asset esista all'interno del ledger, in caso negativo ritornerà una eccezione, altrimenti viene restituito l'intero asset al client.

```
async UpdateAsset(ctx, numTell, ownerRoaming, dateRoaming, timeStartRoaming, seconds, serviceType) {
  var key = producerKey(numTell, dateRoaming, timeStartRoaming);
  const exists = await this.AssetExists(ctx, key);
  if (!exists) {
    throw new Error(`The asset ${key} does not exist`);
  }
  const updatedAsset = {
    NumTell: numTell,
    OwnerRoaming: ownerRoaming,
    DateRoaming: dateRoaming,
    TimeStartRoaming: timeStartRoaming,
    Seconds: seconds,
    ServiceType: serviceType
  };
  return ctx.stub.putState(key, Buffer.from(JSON.stringify(updatedAsset)));
}
```

Figura 23. Metodo per aggiornare un asset presente nel ledger

Una volta che abbiamo degli asset nel ledger con cui possiamo interagire, il metodo illustrato nella *Figura 23*, ovvero UpdateAsset, permette di aggiornare gli attributi dell'asset richiesto. Per prima cosa, viene verificato che l'asset richiesto esista, in caso negativo viene tornata una eccezione, altrimenti viene modificato tale l'asset ed inserito nel ledger, quest'ultima operazione viene effettuata interagendo con l'oggetto stub.

```
async DeleteAsset(ctx, numTell) {
  const exists = await this.AssetExists(ctx, numTell);
  if (!exists) {
    throw new Error(`The asset ${numTell} does not exist`);
  }
  return ctx.stub.deleteState(numTell);
}
```

Figura 24. Metodo per eliminare un asset dal ledger

Potrebbero esserci casi in cui si ha il bisogno della possibilità di eliminare un asset dal ledger, in questo caso viene illustrato il metodo nella *Figura 24*, ovvero DeleteAsset, per gestire tale requisito. Questo metodo, verifica prima che l'asset richiesto è presente nel ledger, in caso negativo ritorna una eccezione, altrimenti, interagendo con l'oggetto stub, elimina tale asset dal ledger.

```
async AssetExists(ctx, numTell) {
  const assetJSON = await ctx.stub.getState(numTell);
  return assetJSON && assetJSON.length > 0;
}
```

Figura 25. Metodo che verifica se esiste una dato asset nel ledger

Può essere una buona idea controllare se un asset esista prima di eseguire una determinato operazione, ad esempio può essere utilizzata prima di creare un nuovo asset per verificare se quest'ultimo da inserire è realmente “nuovo”. E quindi nella *Figura 25* viene illustrata la funzione per tale utilizzo, ovvero AssetExists.

```
async GetAllAssets(ctx) {
  const allResults = [];
  const iterator = await ctx.stub.getStateByRange('', '');
  let result = await iterator.next();
  while (!result.done) {
    const strValue = Buffer.from(result.value.value.toString()).toString('utf8');
    let record;
    try {
      record = JSON.parse(strValue);
    } catch (err) {
      console.log(err);
      record = strValue;
    }
    allResults.push({ Key: result.value.key, Record: record });
    result = await iterator.next();
  }
  return JSON.stringify(allResults);
}
```

Figura 26. Metodo che restituisce tutti gli asset contenuti nel ledger

Infine, nella *Figura 26*, viene illustrato il metodo, ovvero GetAllAssets, che permette l'interrogazione del ledger per restituire tutti gli asset presenti al suo interno. Per facilitare tale operazione viene utilizzato un iteratore che permette di tenere traccia di tutti i record di dati, per poi restituirli al client.

E con questa funzione si conclude la spiegazione dello script dello smart contract. Per quanto riguarda gli script dei 2 client (o applicazioni) questi sono molto simili, cambiano solamente le operazioni che effettuano nella rete.

```
// build an in memory object with the network configuration (also known as a connection profile)
const ccp = buildCCPOrg1();

// build an instance of the fabric ca services client based on
// the information in the network configuration
const caClient = buildCAClient(FabricCAServices, ccp, 'ca.org1.example.com');

// setup the wallet to hold the credentials of the application user
const wallet = await buildWallet(Wallets, walletPath);

// in a real application this would be done on an administrative flow, and only once
await enrollAdmin(caClient, wallet, mspOrg1);
```

Figura 27. Registrazione amministratore dell'applicazione

Innanzitutto, l'applicazione registra l'utente amministratore, da notare che la registrazione dell'utente sono interazioni che avvengono tra l'applicazione e la Certificate Authority, non tra l'applicazione e il chaincode. Infatti, lo script dello smart contract non contiene alcuna funzionalità che supporti la registrazione dell'amministratore, né tanto meno quella dell'utente.

Nel codice dell'applicazione illustrato nella *Figura 27*, dopo aver ottenuto il riferimento al percorso del profilo di connessione comune, assicurando che il profilo di connessione esista e specificando dove creare il wallet che contiene le identità di rete, il metodo enrollAdmin() viene eseguito e le credenziali di amministratore vengono generate dalla Certificate Authority. Questo metodo memorizza le credenziali dell'amministratore generate dalla CA nel wallet.

```
// in a real application this would be done only when a new user was required to be added
// and would be part of an administrative flow
await registerAndEnrollUser(caClient, wallet, mspOrg1, org1UserId, 'org1.department1');
```

Figura 28. Registrazione utente dell'applicazione

In secondo luogo, l'applicazione iscrive e registra un utente dell'applicazione, illustrato nella *Figura 28*. Ora che si hanno le credenziali dell'amministratore in un wallet, l'applicazione utilizza l'admin per iscriverne e registrare un utente dell'applicazione che verrà utilizzato per interagire con la rete blockchain. Questa operazione è simile alla registrazione dell'amministratore, ma utilizza un CSR (Certificate Signing Request) per registrarsi e memorizzare le proprie credenziali insieme a quelle dell'amministratore nel wallet.

```
// Create a new gateway instance for interacting with the fabric network.
// In a real application this would be done as the backend server session is setup for
// a user that has been verified.
const gateway = new Gateway();

try {
  // setup the gateway instance
  // The user will now be able to create connections to the fabric network and be able to
  // submit transactions and query. All transactions submitted by this gateway will be
  // signed by this user using the credentials stored in the wallet.
  await gateway.connect(ccp, {
    wallet,
    identity: org1UserId,
    discovery: { enabled: true, asLocalhost: true } // using asLocalhost as this gateway is using a fabric network deployed locally
  });

  // Build a network instance based on the channel where the smart contract is deployed
  const network = await gateway.getNetwork(channelName);

  // Get the contract from the network.
  const contract = network.getContract(chaincodeName);
```

Figura 29. Preparazione della connessione al canale e allo smart contract

In terzo luogo, l'applicazione prepara una connessione al canale e allo smart contract da utilizzare. Nei passaggi precedenti, l'applicazione ha generato le credenziali dell'amministratore e dell'utente dell'applicazione e le ha inserite nel wallet. Se le credenziali esistono e sono associate agli attributi di autorizzazione corretti, l'utente dell'applicazione sarà in grado di chiamare le funzioni chaincode

dopo aver ottenuto il riferimento al nome del canale e al nome dello smart contract. Nella *Figura 29* l'applicazione ottiene il riferimento allo smart contract utilizzando il suo nome, e anche il nome del canale, tramite l'oggetto gateway, che viene utilizzato per interagire con la rete Fabric.

```
// InitLedger:
console.log('\n--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger');
let latencyStart= (new Date()).getTime();
await contract.submitTransaction('InitLedger');
let latencyEnd = (new Date()).getTime();
console.log('*** Result: committed');
console.log('--> The latency time of operations is '+(latencyEnd-latencyStart)+ ' milliseconds');

// GetAllAssets:
console.log('\n--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger');
let latencyStart= (new Date()).getTime();
let result = await contract.evaluateTransaction('GetAllAssets');
let latencyEnd = (new Date()).getTime();
console.log('*** Result: ${prettyJSONString(result.toString())}');
console.log('--> The latency time of operations is '+(latencyEnd-latencyStart)+ ' milliseconds');
```

Figura 30. Operazioni effettuate dalla prima applicazione

```
// CreateAsset n.1
let roaming = Math.floor(Math.random() * 9999);
console.log('\n--> Submit Transaction: CreateAsset n.1, creates new asset with 3307043222, Operatore2, '+date+', '+time+', '+roaming+' and internet arguments');
let latencyStart= (new Date()).getTime();
let result = await contract.submitTransaction('CreateAsset', '3307043222', 'Operatore2', date, time, roaming, 'internet');
let latencyEnd = (new Date()).getTime();
console.log('*** Result committed: ${prettyJSONString(result.toString())}');
console.log('--> The latency time of operations is '+(latencyEnd-latencyStart)+ ' milliseconds');

//...

// GetAllAssets:
console.log('\n--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger');
let latencyStart= (new Date()).getTime();
let result = await contract.evaluateTransaction('GetAllAssets');
let latencyEnd = (new Date()).getTime();
console.log('*** Result: ${prettyJSONString(result.toString())}');
console.log('--> The latency time of operations is '+(latencyEnd-latencyStart)+ ' milliseconds');
```

Figura 31. Operazioni effettuate dalla seconda applicazione

Infine, le applicazioni sono pronte per richiamare tutte le funzionalità dello smart contract descritto precedentemente. Nelle *Figure 30* e *31* vengono effettuate una serie di operazioni, alcune vengono eseguite dal primo applicativo e altre dal secondo applicativo. Più precisamente, il primo client effettua una inizializzazione del ledger con alcuni asset per poi stampare quest'ultimi, mentre, il secondo client crea ed inserisce nuovi asset all'interno del ledger per poi stampare alla fine sia gli asset dell'inizializzazione, inseriti dal primo client, che i nuovi asset creati dal

secondo client. Da notare che la funzione `submitTransaction()` viene utilizzata per richiamare le funzioni contenute nel `chaincode` per interagire col `ledger`. In realtà, la funzione `submitTransaction ()` utilizza uno specifico servizio per trovare un set di peer di approvazione richiesti per il `chaincode`, invocarlo sul numero richiesto di peer, raccogliere i risultati approvati da quei peer e infine inviare la transazione all'`ordering service`. Infine, vengono anche calcolati i tempi di latenza di ogni operazione, questi dati serviranno più avanti per effettuare una valutazione del sistema proposto. Tale valutazione si trova nell'apposita sezione nel capitolo successivo.

CAPITOLO 6

6. SISTEMA IN ESECUZIONE

In questo capitolo viene eseguito il sistema proposto del capitolo precedente.

Nella prima sezione viene illustrata l'esecuzione del sistema che realizza la rete federated operators.

Nella seconda sezione viene trattata la valutazione riguardante il sistema in esecuzione.

Nella terza sezione vengono illustrate dei possibili miglioramenti al sistema.

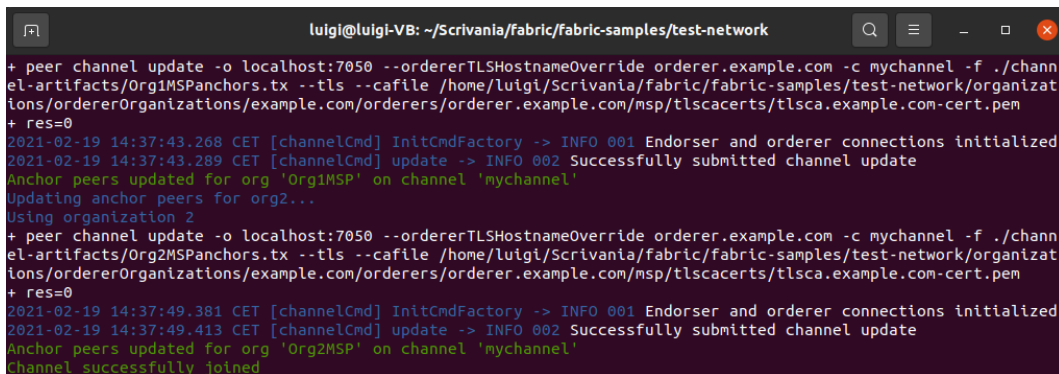
E per finire, nella quarta sezione vengono tratte le conclusioni di questo lavoro di tesi.

6.1 ESECUZIONE

In questa sezione vengono illustrati tutti gli screenshot relativi ai terminali aperti per l'esecuzione del sistema e l'output che questi restituiscono.

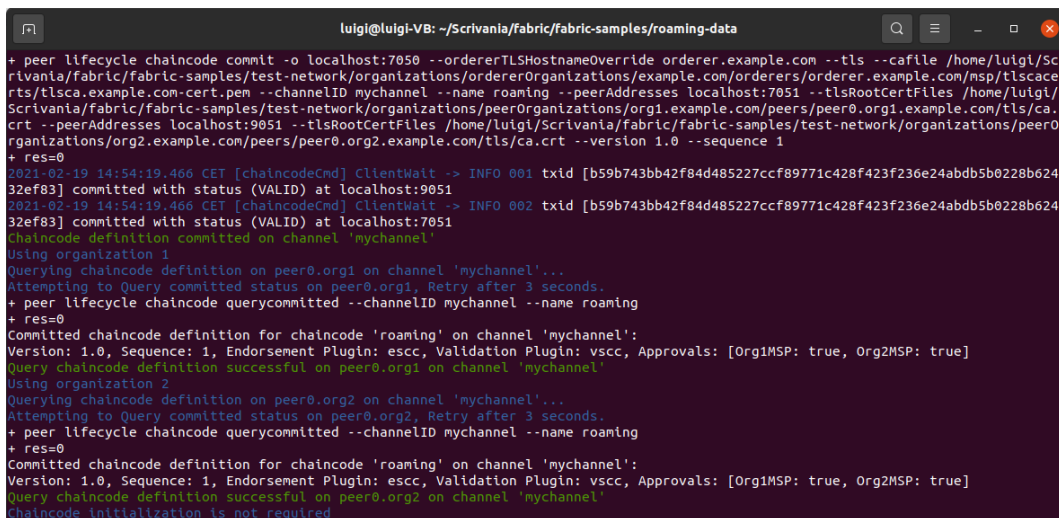
Come detto nel capitolo precedente, la rete di tipo federated operators, in questa tesi, viene denominato "roaming-data" e viene utilizzata la tecnologia Hyperledger Fabric. Per eseguire tale sistema ci sono tre terminali aperti, ciascuno per un componente specifico. In più, all'interno della cartella "roaming-data", sono presenti vari script con lo scopo di semplificare la creazione della rete, la distribuzione del chaincode, l'avvio delle applicazioni javascript dei client ed altri, come il riavvio e lo shutdown della rete.

6. SISTEMA IN ESECUZIONE



```
luigi@luigi-VB: ~/Scrivania/fabric/fabric-samples/test-network
+ peer channel update -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c mychannel -f ./channel-artifacts/Org1MSPanchors.tx --tls --cafile /home/luigi/Scrivania/fabric/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
+ res=0
2021-02-19 14:37:43.268 CET [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-02-19 14:37:43.289 CET [channelCmd] update -> INFO 002 Successfully submitted channel update
Anchor peers updated for org 'Org1MSP' on channel 'mychannel'
Updating anchor peers for org2...
Using organization 2
+ peer channel update -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c mychannel -f ./channel-artifacts/Org2MSPanchors.tx --tls --cafile /home/luigi/Scrivania/fabric/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
+ res=0
2021-02-19 14:37:49.381 CET [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-02-19 14:37:49.413 CET [channelCmd] update -> INFO 002 Successfully submitted channel update
Anchor peers updated for org 'Org2MSP' on channel 'mychannel'
Channel successfully joined
```

Figura 32. Avvio della rete e creazione canale

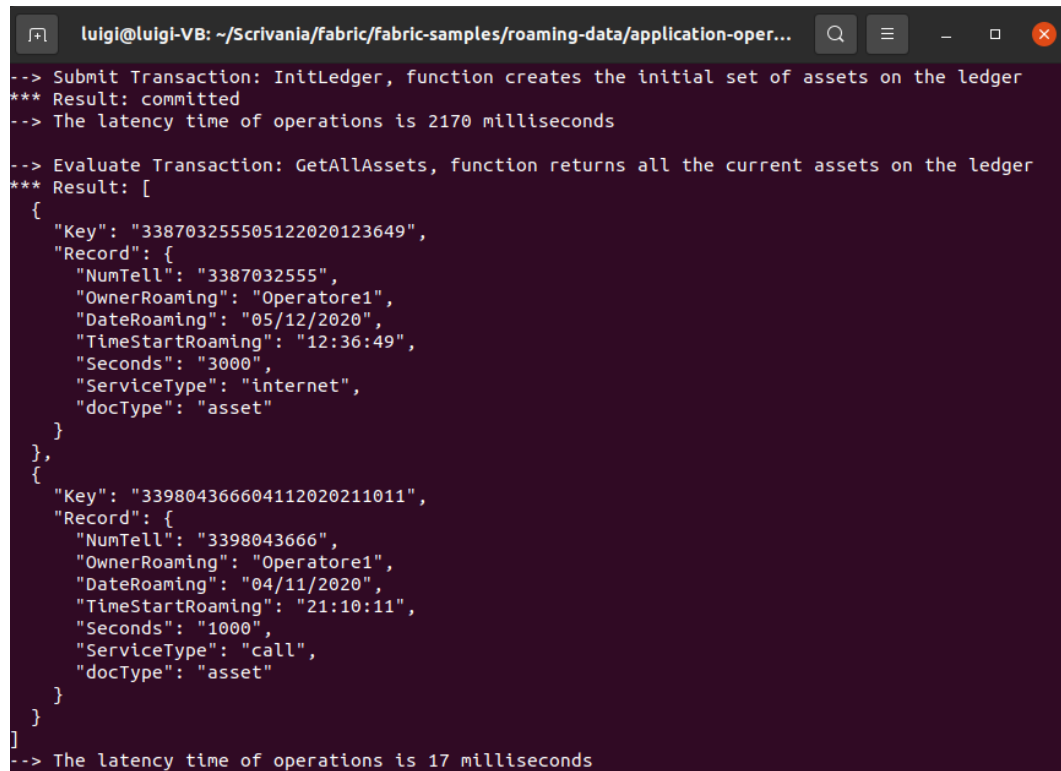


```
luigi@luigi-VB: ~/Scrivania/fabric/fabric-samples/roaming-data
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/luigi/Scrivania/fabric/fabric-samples/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychannel --name roaming --peerAddresses localhost:7051 --tlsRootCertFiles /home/luigi/Scrivania/fabric/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles /home/luigi/Scrivania/fabric/fabric-samples/test-network/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --version 1.0 --sequence 1
+ res=0
2021-02-19 14:54:19.466 CET [chaincodeCmd] ClientWait -> INFO 001 txid [b59b743bb42f84d485227ccf89771c428f423f236e24abdb5b0228b62432ef83] committed with status (VALID) at localhost:9051
2021-02-19 14:54:19.466 CET [chaincodeCmd] ClientWait -> INFO 002 txid [b59b743bb42f84d485227ccf89771c428f423f236e24abdb5b0228b62432ef83] committed with status (VALID) at localhost:7051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name roaming
+ res=0
Committed chaincode definition for chaincode 'roaming' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escv, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name roaming
+ res=0
Committed chaincode definition for chaincode 'roaming' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escv, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
```

Figura 33. Distribuzione del chaincode sulla rete

Prima di tutto, apriamo il primo terminale e posizioniamoci nella cartella “roaming-data” per effettuare il build della rete. Come già detto, ci sono alcuni script utili, tra cui uno chiamato “start.sh”, che effettua l’avvio della rete di test, quindi eseguiamolo dal terminale col comando “./start.sh”. In realtà questo script effettua anche la distribuzione del chaincode sulla rete. Pertanto, viene fatto prima il build della rete, dove vengono stampati sul terminale i messaggi nella *Figura 32*, e poi distribuito il chaincode, dove vengono stampati sempre sullo stesso terminale i messaggi in *Figura 33*.

Se l'output è come quello nelle figure, allora abbiamo una rete blockchain funzionante contenente una chaincode che permette determinate operazioni viste nel capitolo precedente.



```

luigi@luigi-VB: ~/Scrivania/fabric/fabric-samples/roaming-data/application-oper...
--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger
*** Result: committed
--> The latency time of operations is 2170 milliseconds

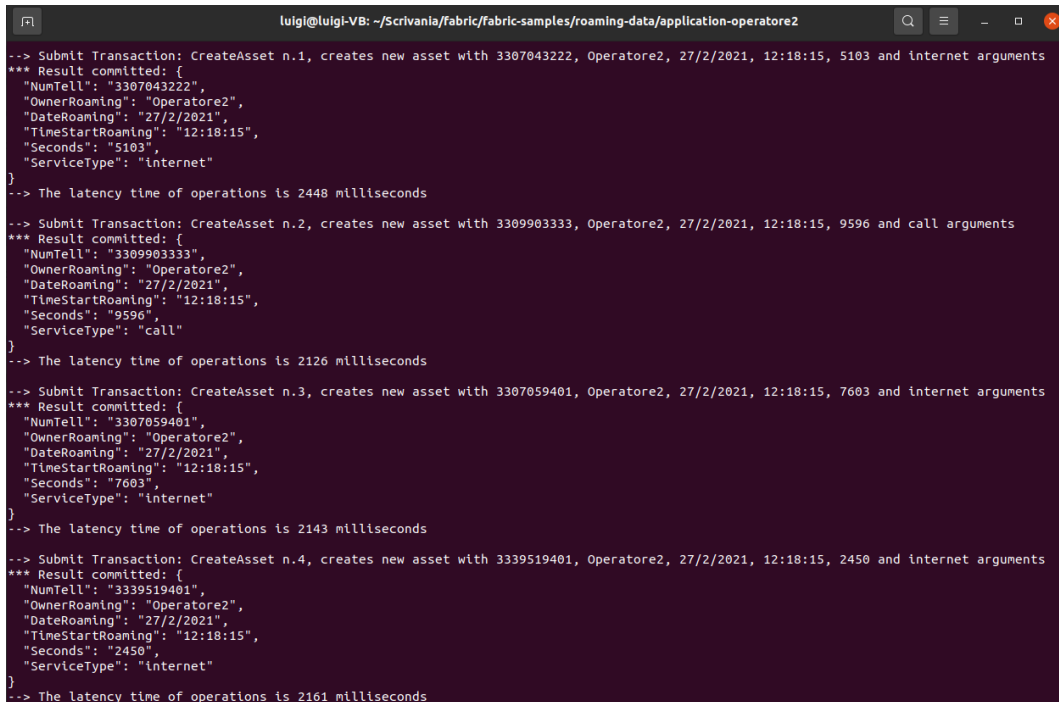
--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger
*** Result: [
  {
    "Key": "338703255505122020123649",
    "Record": {
      "NumTell": "3387032555",
      "OwnerRoaming": "Operatore1",
      "DateRoaming": "05/12/2020",
      "TimeStartRoaming": "12:36:49",
      "Seconds": "3000",
      "ServiceType": "internet",
      "docType": "asset"
    }
  },
  {
    "Key": "339804366604112020211011",
    "Record": {
      "NumTell": "3398043666",
      "OwnerRoaming": "Operatore1",
      "DateRoaming": "04/11/2020",
      "TimeStartRoaming": "21:10:11",
      "Seconds": "1000",
      "ServiceType": "call",
      "docType": "asset"
    }
  }
]
--> The latency time of operations is 17 milliseconds

```

Figura 34. Esecuzione della prima applicazione

Ora possiamo iniziare ad eseguire le due applicazioni descritte nel capitolo precedente. Apriamo un altro terminale e posizioniamoci in un'altra cartella sempre contenuta in “roaming-data”, ovvero “application-operatore1”. Una volta dentro, è presente uno script chiamato “operatore1.sh”, che effettuerà la compilazione e l'avvio dell'applicazione scritta in javascript. Pertanto, nel nuovo terminale aperto, diamo il comando “./operatore1.sh”. Se tutto va a buon fine, e l'output è come quello della *Figura 34*, allora la prima applicazione avrà inizializzato il ledger con alcuni dati iniziali ed avrà effettuato la stampa di quegli stessi valori, nonché i loro tempi di latenza.

6. SISTEMA IN ESECUZIONE



```
luigi@luigi-VB: ~/Scrivania/fabric/fabric-samples/roaming-data/application-operatore2
--> Submit Transaction: CreateAsset n.1, creates new asset with 3307043222, Operatore2, 27/2/2021, 12:18:15, 5103 and internet arguments
*** Result committed: {
  "NumTell": "3307043222",
  "OwnerRoaming": "Operatore2",
  "DateRoaming": "27/2/2021",
  "TimeStartRoaming": "12:18:15",
  "Seconds": "5103",
  "ServiceType": "internet"
}
--> The latency time of operations is 2448 milliseconds

--> Submit Transaction: CreateAsset n.2, creates new asset with 3309903333, Operatore2, 27/2/2021, 12:18:15, 9596 and call arguments
*** Result committed: {
  "NumTell": "3309903333",
  "OwnerRoaming": "Operatore2",
  "DateRoaming": "27/2/2021",
  "TimeStartRoaming": "12:18:15",
  "Seconds": "9596",
  "ServiceType": "call"
}
--> The latency time of operations is 2126 milliseconds

--> Submit Transaction: CreateAsset n.3, creates new asset with 3307059401, Operatore2, 27/2/2021, 12:18:15, 7603 and internet arguments
*** Result committed: {
  "NumTell": "3307059401",
  "OwnerRoaming": "Operatore2",
  "DateRoaming": "27/2/2021",
  "TimeStartRoaming": "12:18:15",
  "Seconds": "7603",
  "ServiceType": "internet"
}
--> The latency time of operations is 2143 milliseconds

--> Submit Transaction: CreateAsset n.4, creates new asset with 3339519401, Operatore2, 27/2/2021, 12:18:15, 2450 and internet arguments
*** Result committed: {
  "NumTell": "3339519401",
  "OwnerRoaming": "Operatore2",
  "DateRoaming": "27/2/2021",
  "TimeStartRoaming": "12:18:15",
  "Seconds": "2450",
  "ServiceType": "internet"
}
--> The latency time of operations is 2161 milliseconds
```

Figura 35. Esecuzione della seconda applicazione

Infine, eseguiamo la seconda applicazione, e per farlo, apriamo un altro terminale e posizioniamoci in un'altra cartella sempre contenuta in “roaming-data”, ovvero “application-operatore2”. In questa cartella, come in quella precedente, c'è uno script chiamato “operatore2.sh”, che ha lo stesso funzionamento del precedente, e quindi diamo il comando “./operatore2.sh” nel terminale. Se va tutto bene, e l'output è come quello della *Figura 35*, allora la seconda applicazione avrà creato degli ulteriori asset e inseriti all'interno del ledger, fatto ciò, effettuerà la stampa di tutti gli asset contenuti nel ledger, e come la prima applicazione, dopo ogni operazione stamperà anche i tempi di latenza.

Ora che tutto è stato eseguito correttamente, possiamo vedere il sistema in esecuzione, il quale fornisce un esempio di scambio di asset tra due operatori che fanno parte della stessa rete blockchain, ovvero un esempio di rete federated operators.

6.2 VALUTAZIONE

Ora che il sistema proposto è stato spiegato nel dettaglio, con implementazione ed esecuzione, è possibile fornire una valutazione globale. Pertanto, si può passare ad analizzare il tempo di latenza necessario per effettuare un trasferimento di asset nella stessa rete, e quindi per lo scenario federated operators.

Il sistema è stato eseguito su una macchina virtuale in locale. Prima di tutto, come si evince nella *Figura 34* della sezione precedente, notiamo che l'operazione di lettura degli asset nel ledger è abbastanza immediata, con tempo di circa 15 millisecondi. Il tempo più interessante è quello inerente alla creazione di un asset. Infatti, come si vede nella *Figura 35* (ma anche nella *Figura 34* con l'operazione di inizializzazione) con le operazioni di creazione degli asset, il tempo è più alto.

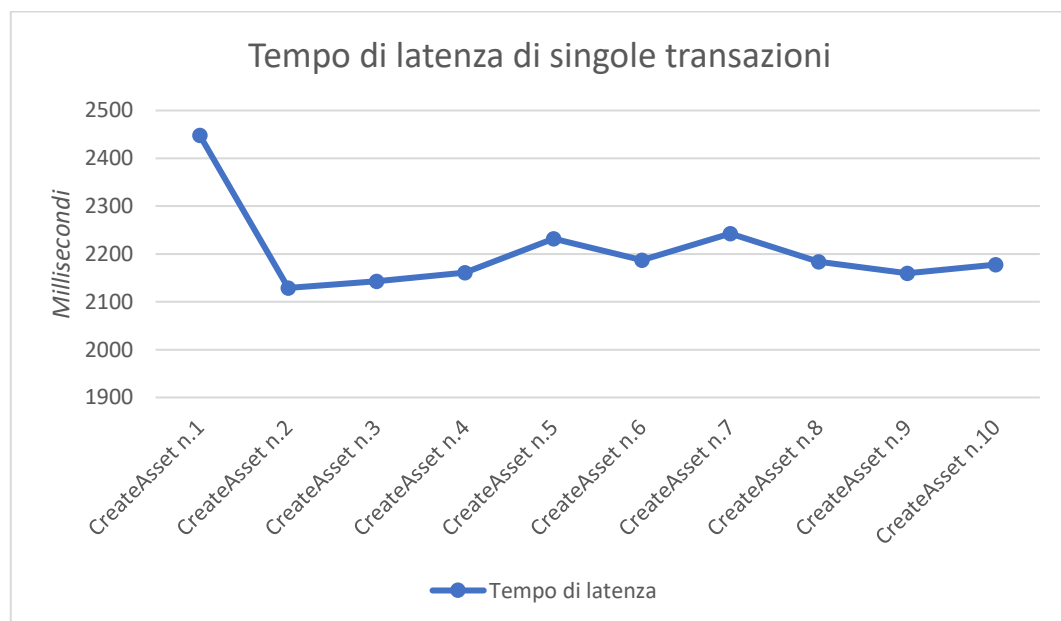


Figura 36. Grafico tempo di latenza delle transazioni interne

Come illustrato dal grafico, nella *Figura 36*, la creazione di un nuovo asset e il suo successivo inserimento nel ledger, ha un tempo quasi costante con una media di 2206 millisecondi, ovvero 2 secondi. Nel grafico vengono eseguite delle singole transazioni in sequenza, in tutto 10 transazioni.

Di seguito, invece, viene illustrato un altro grafico che mostra i tempi di latenza di un insieme di transazioni e non singole transazioni come il precedente.

```
--> The latency time of 1 transaction is 2186 milliseconds
--> The latency time of 5 transactions is 10490 milliseconds
--> The latency time of 10 transactions is 20864 milliseconds
--> The latency time of 15 transactions is 31285 milliseconds
--> The latency time of 20 transactions is 41538 milliseconds
--> The latency time of 25 transactions is 52020 milliseconds
--> The latency time of 30 transactions is 62244 milliseconds
```

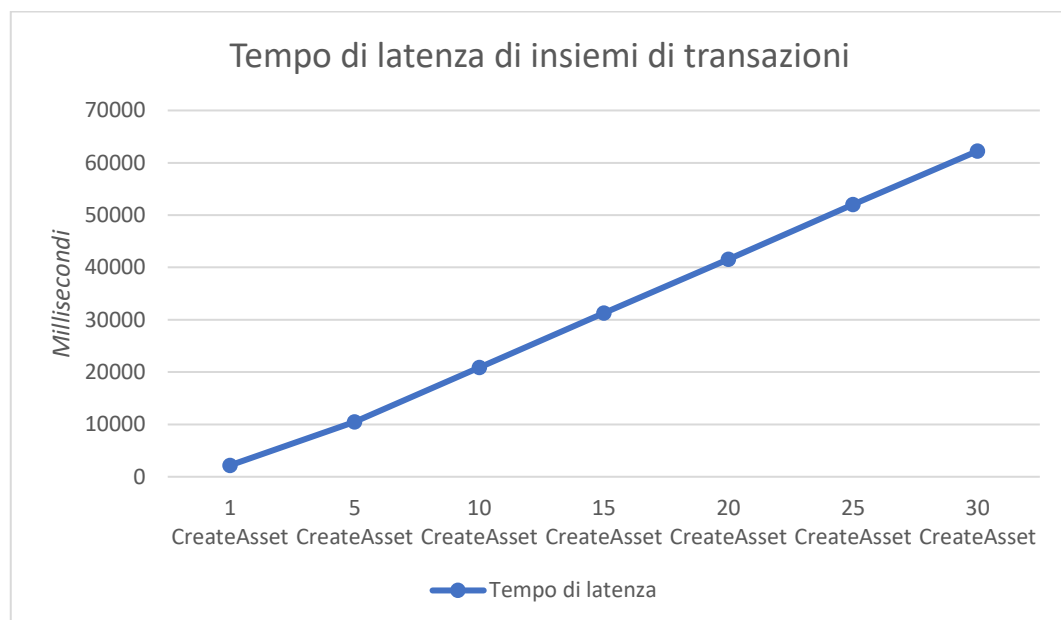


Figura 37. Grafico tempo di latenza di insiemi di transazioni

Come illustrato dal grafico, nella *Figura 37*, la creazione di un insieme di asset ha un proprio tempo. Più precisamente, vediamo dal grafico che ogni 5 transazioni il tempo aumenta di circa 10 secondi.

Lo scenario, federated operators, è stato implementato anche con un'altra tecnologia blockchain, ovvero quella di tipo Hyperledger Sawtooth. Questa nuova implementazione è stata fatta con lo scopo di fornire un confronto tra i tempi di latenza tra due blockchain eterogenee.

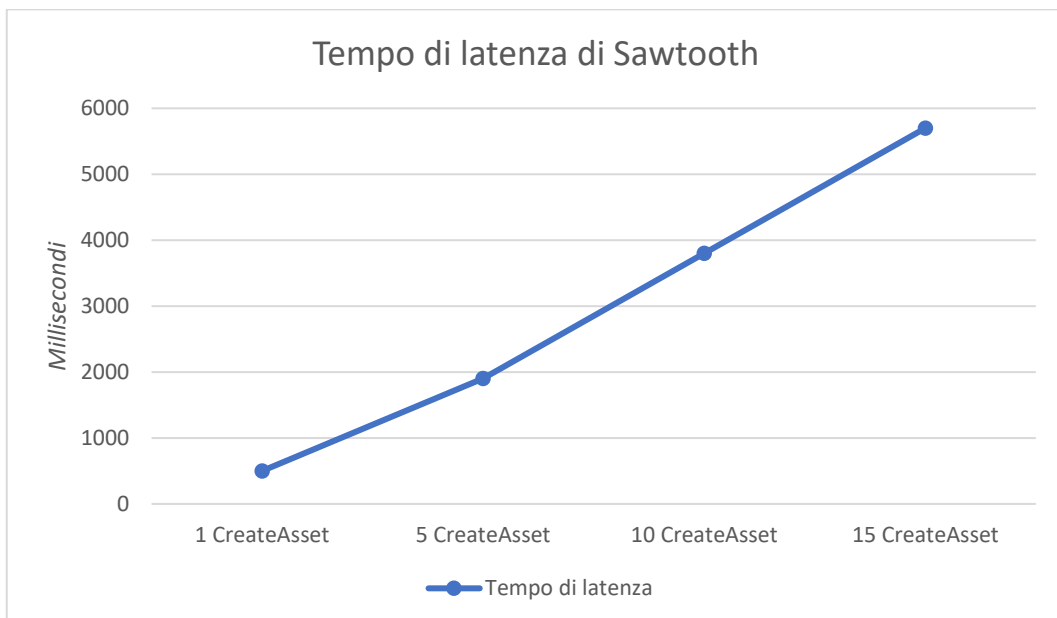


Figura 38. Grafico tempo di latenza di Sawtooth

Come illustrato dal grafico, nella *Figura 38*, tra la tecnologia Hyperledger Fabric e Sawtooth i tempi di latenza sono molto diversi. Infatti, Sawtooth impiega circa 1900 millisecondi, ovvero circa 2 sec, per effettuare 5 transazioni, mentre Fabric ne impiega circa 10 di secondi. In conclusione, anche se non sono state utilizzate le stesse macchine, ad un primo sguardo, la tecnologia Sawtooth è migliore.

6.3 MIGLIORAMENTI FUTURI

In questa sezione viene illustrato un possibili miglioramento che può essere applicato, aggiungere funzionalità al sistema.

Si può sfruttare l'idea dell'interoperabilità tra blockchain per implementare uno scenario not-federated operators, ovvero degli operatori che non fanno parte della stessa rete e che usano una tecnologia blockchain differente. Ad esempio, un primo operatore utilizza Hyperledger Fabric, mentre un secondo operatore utilizza Hyperledger Sawtooth, ed entrambi vogliono scambiare asset tra loro. Ovviamente una prima difficoltà è che le due tecnologie blockchain sono differenti e per tanto hanno logica e infrastrutture leggermente diverse, ma sfruttando l'interoperabilità tra blockchain si potrebbe cercare di metterli in comunicazione, stando attenti a conservare le principali proprietà delle blockchain, ovvero indipendenza di terze parti, scalabilità e sicurezza.

6.4 CONCLUSIONE

In conclusione, la visione blockchain per le reti di telecomunicazione consente a più applicazioni e servizi di funzionare su un livello comune di identità, logica aziendale e governance. Gli operatori possono implementare una o più infrastrutture blockchain che possono effettuare transazioni per vari scopi mantenendo una forte privacy. Dal punto di vista degli operatori, con l'integrazione delle blockchain, possono fare a meno di terze parti che aumentano il tempo di latenza di una transazione ma più importante risolvono il problema della fiducia. La sicurezza della rete è nettamente superiore grazie alle blockchain private, dove i vari nodi

sono autorizzati a partecipare alla rete e le informazioni inserite nel ledger sono immutabili.

In questa tesi sono state presentate le problematiche dei sistemi di roaming attuali, nel capitolo 2, il concetto di blockchain e i vantaggi dell'integrazione nei sistemi di roaming, nel capitolo 3. È stata approfondita la tecnologia blockchain, ovvero Hyperledger Fabric, nel capitolo 4. Sono state illustrate le idee principali e le procedure di roaming di base basate sugli smart contract, nel capitolo 5. Inoltre, è stato fornito un confronto tra due DLT differenti che implementano lo stesso scenario, analizzando i tempi di latenza delle operazioni, nel capitolo 6.

BIBLIOGRAFIA

- [1] G. Macia-Fernandez, P. Garcia-Teodoro and J. Diaz-Verdejo, "Fraud in roaming scenarios: an overview," in IEEE Wireless Communications, vol. 16, no. 6, pp. 88-94, December 2009, doi: 10.1109/MWC.2009.5361183.
<https://ieeexplore.ieee.org/document/5361183>

- [2] Role of Clearinghouses (Billing and Settlement):
<http://what-when-how.com/roaming-in-wireless-networks/role-of-clearinghouses-billing-and-settlement/>

- [3] Weerasinghe, Nisita & Hewa, Tharaka & Dissanayake, Maheshi & Ylianttila, Mika & Liyanage, Madhusanka. (2021). Blockchain-based Roaming and Offload Service Platform for Local 5G Operators.

- [4] A. Refaey, K. Hammad, S. Magierowski and E. Hossain, "A Blockchain Policy and Charging Control Framework for Roaming in Cellular Networks," in IEEE Network, vol. 34, no. 3, pp. 170-177, May/June 2020, doi: 10.1109/MNET.001.1900336.
<https://ieeexplore.ieee.org/document/8910631>

- [5] Nicola Volpe, "Blockchain Interoperability Empowered by Using Trusted Execution Environments"

- [6] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi and J. Wang, "Untangling Blockchain: A Data Processing View of Blockchain Systems," in IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 7, pp. 1366-1385, 1 July 2018, doi: 10.1109/TKDE.2017.2781227.
<https://ieeexplore.ieee.org/abstract/document/8246573>

- [7] Mauro Bellini. Blockchain: cos'è, come funziona e gli ambiti applicativi in Italia
<https://www.blockchain4innovation.it/esperti/blockchain-perche-e-cosi-importante/>

- [8] Hyperledger Fabric
<https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html>

- [9] Blockchain network
<https://hyperledger-fabric.readthedocs.io/en/latest/network/network.html>

- [10] Identity
<https://hyperledger-fabric.readthedocs.io/en/latest/identity/identity.html>

- [11] Membership Service Provider (MSP)
<https://hyperledger-fabric.readthedocs.io/en/latest/membership/membership.html>

- [12] Policies
<https://hyperledger-fabric.readthedocs.io/en/latest/policies/policies.html>

- [13] Peers
<https://hyperledger-fabric.readthedocs.io/en/latest/peers/peers.html>

- [14] Ledger
<https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html>

- [15] The Ordering Service
https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html

- [16] Smart Contracts and Chaincode
<https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html>