Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Medieninformatik

Bachelorarbeit

von

Philipp **Stangl**

Entwurf und Implementierung eines heterogenen Blockchain-Konsortiums für ein Lebensmittel-Lieferketten-Netzwerk

Design and Implementation of a Heterogeneous
Blockchain Consortium for a Food Supply Chain Network

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Medieninformatik

Bachelorarbeit

von

Philipp **Stangl**

Entwurf und Implementierung eines heterogenen Blockchain-Konsortiums für ein Lebensmittel-Lieferketten-Netzwerk

Design and Implementation of a Heterogeneous Blockchain Consortium for a Food Supply Chain Network

Bearbeitungszeitraum:	von	1. September 2021
	bis	31. Januar 2022

1. Prüfer:	Prof. Dr.-Ing. Christoph Neumann
2. Prüfer:	Prof. Dr. rer. nat. Daniel Loebenberger

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Bestätigung gemäß § 12 APO

Name und Vorname der Studentin/des Studenten:	**Stangl, Philipp**
Studiengang:	**Medieninformatik**

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

Entwurf und Implementierung eines heterogenen Blockchain-Konsortiums für ein Lebensmittel-Lieferketten-Netzwerk

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 31. Januar 2021

Unterschrift:

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Bachelorarbeit Zusammenfassung

Studentin/Student (Name, Vorname):	**Stangl, Philipp**
Studiengang:	Medieninformatik
Aufgabensteller, Professor:	Prof. Dr.-Ing. Christoph Neumann
Durchgeführt in (Firma/Behörde/Hochschule):	OTH Amberg-Weiden
Ausgabedatum: 1. September 2021	Abgabedatum: 31. Januar 2022

Titel:

Entwurf und Implementierung eines heterogenen Blockchain-Konsortiums für ein Lebensmittel-Lieferketten-Netzwerk

Zusammenfassung:

In einem Lebensmittel-Lieferketten-Netzwerk befindet sich jedes Unternehmen in einer Netzebene und gehört zu mindestens einer Lieferkette. Das bedeutet, dass eine Organisation gleichzeitig und im Laufe der Zeit an mehreren Lieferketten teilnehmen kann. Die derzeitigen Lösungen für Blockchain-fähige Lieferketten konzentrieren sich auf eine einzelne Blockchain. Dies erfordert speziell für Unternehmen in Lieferketten-Netzwerken ihre Daten auf mehreren Blockchains zu teilen (eine Blockchain für jede Lieferkette). Außerdem haben die Unternehmen keine Kontrolle über die Datentransparenz, das insbesondere beim Austausch sensitiver Daten über eine gemeinsame Blockchain deutlich wird. Im Rahmen dieser Arbeit wird eine Lösung vorgestellt, die es Unternehmen ermöglicht mit ihrer eigenen Blockchain (worauf Unternehmen ihre unveränderlichen Daten speichern können) einer Konsortium Blockchain beizutreten. Mechanismen zur Ermöglichung von Blockchain Interoperabilität tragen dazu bei, die Vorteile unabhängiger, souveräner Blockchains zu bewahren und ermöglichen gleichzeitig die gemeinsame Nutzung von Daten über Blockchain-Grenzen hinweg. Eine Referenzimplementierung beweist die Machbarkeit dieses Konzepts.

Schlüsselwörter: Blockchain, Interoperabilität, Lieferketten Netzwerk

Abstract

In a food supply chain network, each organization is positioned in a network layer and belongs to at least one supply chain. That means an organization may participate in multiple supply chains at the same time and over time. Current solutions for blockchain-enabled supply chains are centered around one standalone blockchain. This requires organizations to share their data on multiple blockchains (one blockchain for each supply chain). Furthermore, organizations have no control over data transparency. In the context of this thesis, a solution is presented that allows organizations to join a consortium blockchain with their own blockchain where they can store their immutable data. Mechanisms for enabling blockchain interoperability help preserve the benefits of independent sovereign blockchains while allowing for sharing of data across blockchain boundaries. A prototype implementation proves the feasibility of the concept.

Contents

Part I

Foundations

Chapter 1

Introduction

The food industry comprises companies dedicated to manufacturing and processing raw materials and semi-finished products from agriculture, forestry, and fishing. In recent years, food supply chains have progressed from shorter, independent to more unified, coherent relationships among supply chain participants [5]. Developing long-term, and collaborative relationships requires evolutionary technological solutions to simultaneously retain a competitive edge.

Blockchain technology is presented as a way to reduce fraud, increase supply chain visibility, and provide supply chain optimization. Current applications of blockchain technology in food supply chain management (e.g., IBM Food Trust) rely mainly on a single distributed ledger. The implications on supply chain networks are twofold: (i) organizations participating in multiple supply chains must share their data on multiple blockchains, and (ii) participants may see information originally not intended for them because all participants can view every transaction on a distributed ledger.

A multi-chain[1] approach is required, allowing organizations to store immutable data on their own blockchain. A decentralized hub coordinates the cross-chain exchange of digital assets among the heterogenous blockchains. The hub further ensures that all parties comply to the overarching rules of the consortium.

This bachelor's thesis presents a solution that supports interoperability and controlled transparency in a heterogeneous, blockchain-enabled supply chain network. It enables divergent types of consensus systems to interoperate in a decentralized federation, allowing public and private blockchains to have controllable access to each other.

[1]The term multi-chain is separated by a hyphen because an enterprise blockchain solution called Multichain is available on the market. Multichain is not related in any way to this work.

1.1 Fundamentals

This section introduces definitions regarding blockchain technology and Food Supply Chain Networks (FSCNs). To begin, the definition of blockchain technology as a decentralized ledger for tracking one or more digital assets on a Peer-to-Peer (P2P) network is expanded in section 1.1.1. Section 1.1.2 describes an FSCN in terms of the involved parties, business processes, and product characteristics.

1.1.1 Blockchain Technology

Blockchain technology, introduced by the Bitcoin protocol [26] in 2008, is a Distributed Ledger Technology (DLT)[2]. Today, blockchain is the most commonly used data structure for distributed ledgers [21]. The combination of cryptographic and game-theoretic concepts enables immutable transactions and automatic consensus about its state by the parties involved. This section provides an outline of blockchain-related terms that are essential to understanding the concept.

Blockchain Types

There are three different types of blockchain systems (depicted in figure 1.1) [45]. Public blockchains are considered permissionless because, in principle, everyone can attend the consensus process and read the stored data. The application of public blockchains has several use cases, including cryptocurrencies and document validation. In a consortium blockchain, an elected group of participants is allowed to attend the consensus process. The stored data may be read by selected members or by the public. Supply chain and research environments are two exemplary use cases for this sort of blockchain. In a private blockchain, all participants belong to the same organization, and the public cannot access the system. Two use cases for this final blockchain type are banking and asset ownership. Private and consortium blockchains are considered permissioned blockchains because, in both cases, only a limited group can attend the consensus process.

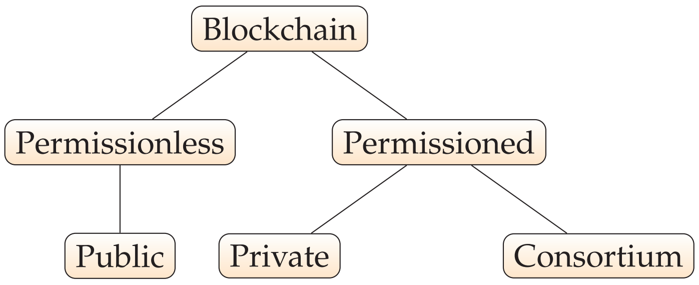


Figure 1.1: Types of blockchain

[2]The terms DLT and blockchain technology are often used interchangeably. However, blockchain is one type of a distributed ledger. Other types of distributed ledgers include Directed Acyclic Graph, Hashgraph, Holochain, and Cerberus.

Blockchain Data Structure

A blockchain denotes a special data structure whose elements, the blocks, are linked together by cryptographic hash functions. Figure 1.2 illustrates that each block is separated into two parts: a body and a header. A block cryptographically binds the body (i.e., a set of transactions) to a header. The block body is used to generate a unique identifier for that block. The block header points to the unique identifier of the preceding block, known as the parent block. Since each block has a pointer to its parent block, the blocks can be deterministically ordered.

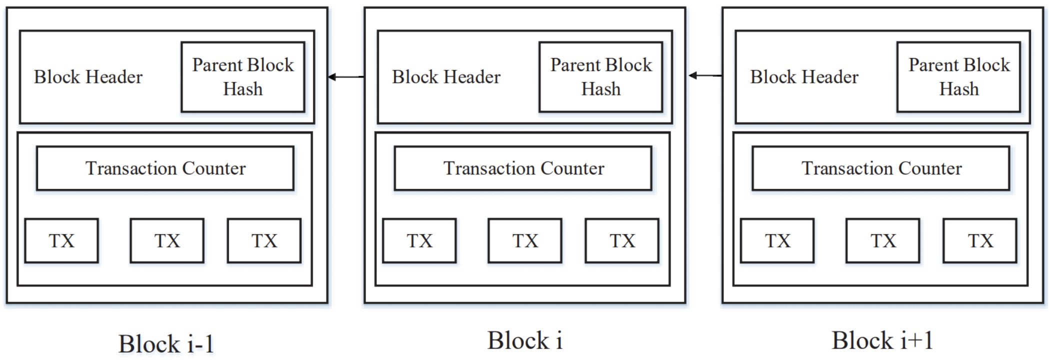


Figure 1.2: Blockchain data structure (adopted from Zheng et al. [45])

Peer-to-Peer Network

The blockchain data structure is distributed across a P2P network (figure 1.3), which is a communications model for decentralized networks. The model consists of a set of devices (called nodes) that collectively store and share files. The P2P architecture of blockchain technology is based on the concept of decentralization. Each node acts as an individual peer. Therefore, communication is conducted without any central administration or server. This means, in theory, that all nodes perform the same tasks and have equal power.

Block Production

The shared ledger's content is not altered by changing existing blocks. Instead, new blocks with instructions on how the ledger state should change from one block to another become appended to the blockchain. These instructions are commonly referred to as transactions. The state transition function defines the rules associated with how the ledger can change. Such rules can either be complex or simple; for instance, users can only exchange assets that they own. Once a valid set of transactions is collected, they are put into the block content. Afterward, the block is placed at the end of the chain. This block production process allows the underlying distributed ledger to change over time.

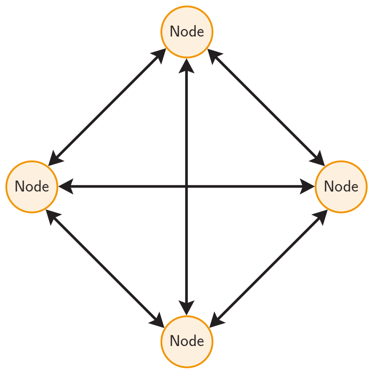


Figure 1.3: Schematic diagram of a P2P network

Block Finalization

After a new block has been produced, it can be shared with others who want to construct the same shared ledger. However, since blockchains are decentralized, it is possible that two different yet still valid blocks compete for the same position at the end of a chain. A block finalization mechanism determines which chain of blocks is the canonical blockchain: for any given blockchain, there should only be one true final state of the shared ledger. Any alternative states of the blockchain are known as forks.

1.1.2 Food Supply Chain Network

A supply chain is an interconnection of organizations, activities, resources, people, and information. Organizations along a food supply chain are dedicated to growing and processing raw materials (e.g., fruits) and semi-finished products (e.g., fruit juices) for delivery to the end customer. Food supply chains are complex and affected by various factors such as the sociopolitical environment [39]. Regulatory bodies such as the US Department of Agriculture (USDA) aim to protect consumer health and increase economic viability. Thus, they release frequent updates to ensure their criteria is met by food supply chains.

In an FSCN, more than one supply chain and more than one business process can be identified, both parallel and sequential in time. The parties involved in the business processes depend on the type of FSCN. This thesis considers an FSCN for fresh agricultural products.

Van der Vorst, Beulens, and Beek have identified farmers, retailers, and their logistics service suppliers as parties involved in an FSCN for fresh agricultural products [39]. Figure 1.4 depicts such a supply chain at the organization level within the context of an FSCN for fresh agricultural products. Each organization is positioned in a product lifecycle stage and belongs to at least one supply chain. That means an organization can have multiple suppliers and customers at the same time and over time. Figure 1.4 visualizes this by showing the perspective of the processor (bold lines), who has multiple connections to distributors and farmers. Other stakeholders such as nongovernmental organizations, governments, and shareholders are indirectly involved at each stage of the product lifecycle.

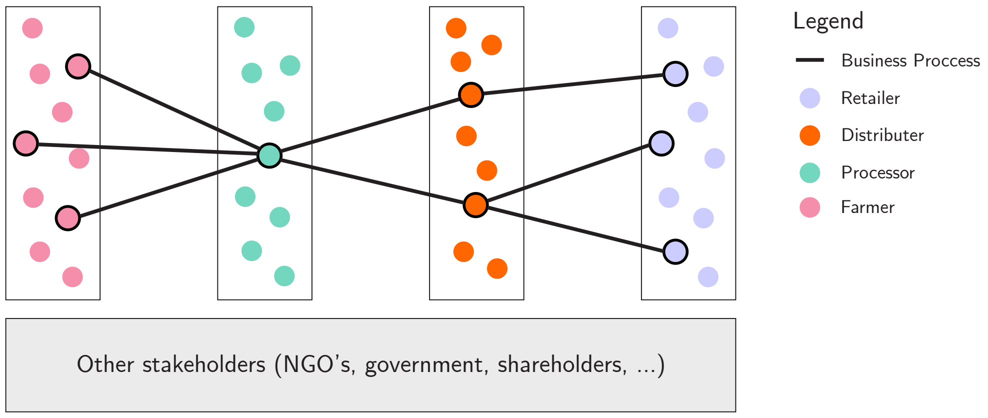


Figure 1.4: Schematic diagram of an FSCN (based on Van der Vorst, Beulens, and Beek [39])

1.2 Motivation

The ability to trace ingredients of any food or product back to their origin in a food supply chain is important. For example, if one is interested in eating organic foods, it could be valuable to know the food's origins. To achieve traceability, end-to-end transparency (i.e., transparency at all stages of a product lifecycle) in a supply chain is required. This transparency is made possible by monitoring each step of the process, capturing all related data, and organizing it in a data storage space.

In today's food supply chains, data is often stored in centralized storage, resulting in organizational silos [29]. As a result, supply chain visibility is limited. Blockchain technology, on the other hand, is built on the premise of decentralization. IBM Food Trust is an example of a blockchain-enabled solution in food supply chains. The underlying distributed ledger is a commercial distribution of Hyperledger Fabric. IBM Food Trust promises organizations to provide a permission-based, shared view of food information with convenient data publishing and controlled sharing of information. In the context of FSCNs, where players participate in multiple supply chains, this would require them to share their data on multiple blockchains. Furthermore, these players collaborate and participate. The state-of-the-art section (2.1) discusses blockchain-enabled supply chain approaches in more detail.

Hyperledger Labs has attempted to reduce the number of parties seeing irrelevant data by isolating different data groups with Private Data Objects (PDOs) [6]. As opposed to raw data access, the access occurs off-ledger in secure enclaves using Intel Software guard extensions. Doing so should preserve data confidentiality and execution integrity and enforce data access policies. The distributed ledger acts as a single authoritative instance, which verifies and records transactions produced by PDOs. This approach adds software complexity. As such, opportunities for weaknesses can emerge to be capitalized on by malicious actors.

1.3 Research Questions

This section clearly defines the scope and direction of this thesis. The first and second research questions are outlined along with the problem statement and objectives.

Problem Statement 1: This thesis identifies a general problem with recording all transactions of a supply chain on a standalone blockchain. The implications are twofold: First, this can lead to varying characteristics of the deployed blockchains. According to a recent American Productivity & Quality Center survey [3], supply chain professionals rate interoperability among the top five blockchain hurdles. This circumstance can become especially challenging in supply chain networks, where a blockchain is required for each supply chain in the network. Secondly, in principle, all transactions can be viewed by all participants who have access to the distributed ledger. Additionally, organizations need flexibility to adapt their blockchain to integrate new technological advancements (for competitive advantage) or to meet changing requirements of regulatory bodies.

Research Question 1 (RQ1): How can a blockchain-enabled supply chain network be designed with interoperability and controlled transparency?

RQ1 Objectives: With ChainFresh, the prototype presented in this thesis, I aim to provide a heterogeneous blockchain consortium that allows organizations (trading partners) to join with their own blockchain, where they store their immutable data. Furthermore, organizations should be able to register shipments on the blockchain and track their journey through the supply chain. Here, the user should be able to monitor a shipment's storage and transportation conditions via a Graphical User Interface (GUI). Additionally, data integration is required to communicate with the external world, such as by accessing and fetching data from an external service. Finally, the application-specific blockchains should be able to interoperate with each other to create cross-chain asset transfers.

Problem Statement 2: The second research question follows from the results of the first research question. Currently, no blockchain evaluation approach is available for a multi-chain environment. Thus, there is no solution available to quantitatively evaluate cross-chain communication, or cross-chain message passing in particular.

Research Question 2 (RQ2): How can cross-chain message passing be quantitatively evaluated in a heterogeneous multi-chain?

RQ2 Objectives: I aim to provide a quantitative evaluation approach for cross-chain messaging under a simulated realistic workload for a running system. This should enable capturing the variability of the transfer volume in cross-chain messaging channels among multiple blockchains. Furthermore, the produced on-chain data must be extracted and processed and the results visualized.

1.4 Thesis Structure

This final section provides an overview of the design process and applied methods used in shaping and implementing the functionality of ChainFresh. In addition to the chronological description of the steps performed, the structure of this thesis is explained in parallel by referring to the corresponding chapter in which a single step is described in detail. Figure 1.5 depicts an overview of this thesis's three-part structure.

Part I, the foundation of this thesis, comprises two chapters. The fundamentals, thesis motivation, research questions, and thesis structure are outlined in the introduction. The primary methodology followed is the three-cycle view of design science research by Hevner [19]. The development of artifacts in each cycle supports the rejection or confirmation of the hypotheses established in chapter 1. Besides the primary method, the current section outlines the auxiliary methods in use to produce the artifacts. Chapter 2 provides a synthesis on state of the art in the research domain through a literature review of related work. The chapter discusses blockchain-enabled supply chain approaches and blockchain interoperability approaches. Ideas from those two domains are constitutive to ChainFresh.

Part II provides an answer to RQ1. Unified Modeling Language (UML) and a system modeling approach are used to compose ChainFresh, as a distributed system, into a three-tier architecture to obtain modules with a well-defined interface and scope of operation. Chapter 3 introduces the ChainFresh concept and provides an overview of the three-tier architecture. The presentation tier is explained in chapter 4, the application tier in chapter 5, and the relay tier in chapter 6. To ease development of the prototype, several frameworks and technologies are used. The substrate framework is employed for the modular blockchain development.

Part III concludes the thesis. Chapter 7 provides a technical evaluation of the Chain-Fresh system. The assessment encompasses the quantitative evaluation for cross-chain messaging, the hard disk footprint of the ChainFresh artifacts, and a discussion about current system limitations. The evaluation approach for cross-chain messaging aims to provide an answer to RQ2 by monitoring the system under a realistic load. Chapter 8 outlines recommendations for future work. Finally, chapter 9 summarizes the results and points to possible implications in practice.

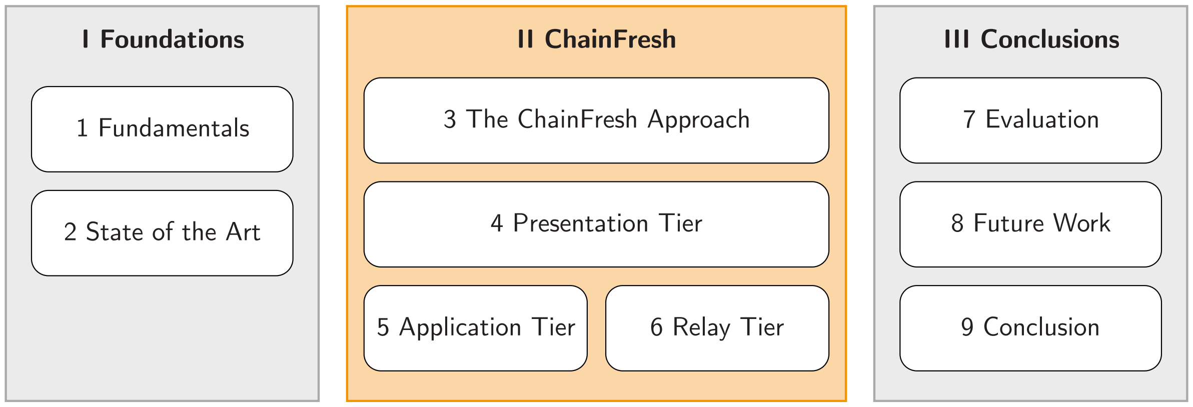


Figure 1.5: Thesis Structure

Chapter 2

State of the Art

This chapter discusses related work explored in the research for ChainFresh. Section 2.1 provides an overview of blockchain-enabled supply chain approaches. Section 2.2 presents the Blockchain Interoperability Framework (BIF), which classifies solutions for blockchain interoperability in three categories, the first two of which are public and hybrid connectors. Section 2.3 focuses on the third category, Blockchain of Blockchains (BoB).

2.1 Blockchain-Enabled Supply Chain

In blockchain-enabled supply chains, information across the product lifecycle is recorded on a ledger for sharing with all supply participants. This can result in faster access to information (e.g., manufacturing information) and thus greater supply chain transparency. Notably, Longo et al. [25] have found that blockchain technology can increase collaboration because it drives competing supply chain companies to share data and information.

This section identifies two categories of blockchain-enabled supply chain approaches. In the literature, there seems to be no general name defined for each identified approach. Therefore, a generic name is derived from the associated related work. Section 2.1.1 discusses the category of solo-chain approaches, wherein all supply chain transactions are recorded on a single blockchain. This category includes the software connector approach, which couples an enterprise information system to a blockchain. In the same category is the smart contract approach, whereby the supply chain management logic is encapsulated in so-called smart contracts. Finally, section 2.1.2 discusses the multi-chain approach that involves multiple domain-specific blockchains.

2.1.1 Solo-Chain Approaches

In the solo-chain approach, transactions happening in the supply chain are recorded on a single blockchain. Those blockchains are typically a copy of an existing open-source protocol or are built from scratch. Either way, the development of individual systems leads to the tendency of varying characteristics; thus, the blockchains of different organizations rarely work well together. An American Productivity & Quality Center survey [3] from 2020 reveals that a lack of interoperability has been rated among the top five blockchain challenges by supply chain professionals.

The most commonly used blockchain for the solo-chain approach is Ethereum. The Ethereum virtual machine is what enables Ethereum to run smart contracts [42], digital contracts containing a collection of data and functions that resides at a specific address on the blockchain. Every time a blockchain participant tries to interact with a smart contract (e.g., executing a transaction that invokes a state-changing function of the smart contract), a transaction fee must be paid. The transaction fee is called *gas* on the Ethereum blockchain. Gas is denoted in the native cryptocurrency Ether [2]. While the cost of gas for each operation is defined in the Ethereum software and is constant, the gas price depends on the number of active blockchain participants.

Smart Contract Approach

An increasingly popular approach to blockchain-enabled supply chains is the use of smart contracts. For instance, Wang et al. [41] have proposed a smart contract-based product traceability system. The system perpetually records all product transferring histories in a distributed ledger by using smart contracts forming a chain that can trace back to the source of the products. Hasan et al. [17] have proposed a smart contract-based supply chain management solution. The proposed solution utilizes Ethereum smart contracts to manage shipment conditions, automate payments, legitimize the receiver, and issue a refund in case of violations to predefined shipping conditions (e.g., temperature, humidity, pressure).

In both cases, this approach appears to have a major drawback for businesses. One cannot update the smart contract (except for its name) once the contract is deployed on the network. In general, organizations may not risk deploying faulty smart contracts that are immutable and suffer the irreversible consequences of their automatic enforcement. Moreover, smart contracts rely on their blockchain core ledger for real-world data integration, which typically utilizes oracles. A blockchain oracle is an external service that queries, verifies, and authenticates external data sources. It typically listens to blockchain events. In the case of an event, it relays the requested information. Since oracles are third-party services, they might have several flaws regarding security, scalability, and infrastructure efficiency.

Software Connector Approach

Longo et al. have presented a software connector to connect an Ethereum-like public blockchain with an enterprise information system [25]. The software connector allows companies to share information with their partners with different levels of visibility. This is achieved similarly to PDOs with visibility groups (i.e., the companies that are authorized to view their data). The supply chain data is stored in an off-chain data storage. It is all the information regarding demand forecasts and inventory levels.

Then a smart contract on the blockchain publishes only a unique key (hash sum) that maps to the data on-chain.

2.1.2 Multi-Chain Approach

Schulz and Freund [31] have proposed a blockchain-enabled distributed supply chain. Their main idea is a network-centric design, which incorporates domain-specific blockchains for handling specific business process and a hub or main blockchain that connects the blockchains to communicate with each other. Schulz and Freund have reasoned, "Systems big enough to cope with a large amount of data, fast enough to cope with a high frequency of transactions, specialized to be extremely secure for certain purposes or to be easily accessible for customers have to be split up in subsystems that need one hub, bringing them all together" (ch. 4, p. 281). To construct such a system, Schulz and Freund have identified three essential pillars: (i) a P2P network with client nodes that are paired with a user interface and permissions, (ii) a file storage either on the blockchain or on a P2P file system (e.g., InterPlanetary File System), and (iii) blockchain interoperability to enable the domain-specific blockchains to communicate with each other.

2.2 Blockchain Interoperability Framework

A limiting factor of the solo-chain approach is that resources are strictly confined to the underlying distributed ledger. Blockchain interoperability involves the ability of independent distributed ledger networks to communicate with each other. Various approaches have been established to provide blockchain interoperability, resulting in a highly fragmented market [4]. Belchior et al. were the first to conduct a systematic literature review on blockchain interoperability solutions. Their main contribution is the BIF, a framework that aims to provide a holistic overview through analyzing the solution space by asking the following six questions: Who controls the cross-chain transaction process?; What assets are exchanged?; Where are the assets transferred from and transferred to (i.e., what are the source and target ledgers)?; When is the execution of cross-chain transactions defined: at design time or run time?; and How are cross-chain transactions realized on the underlying distributed ledgers? Furthermore, the survey classifies studies into three main categories: public connectors, hybrid connectors, and BoB.

This section presents the concepts of public and hybrid connectors for consortium blockchains. The next section (2.3) is dedicated only to the discussion of the BoB due to its primary importance for this work.

2.2.1 Public Connectors

The public connectors category identifies blockchain interoperability approaches across public blockchains. The first type of public connectors is *sidechains*. Here, one blockchain (the main chain) considers another blockchain as an extension of itself (the sidechain). The main chain maintains the distributed ledger of assets and is connected to one or multiple sidechains. A sidechain is used to offload transactions from the main chain. The sidechain processes the offloaded transactions and redirects the outcome back to the main chain. To achieve this, the main chain and sidechain must communicate with each other using an appropriate cross-chain communication protocol. Sidechains are not suitable for a multi-chain approach (section 2.1.2) because they are not independent blockchains.

Other solutions in this category include *notary scheme* and *hashed time-locks contracts*. Both solutions are intended for the exchange of cryptocurrencies. Thus, they are not suitable for an application in supply chains.

2.2.2 Hybrid Connectors

The hybrid connector category is composed of blockchain interoperability approaches that neither fit into the public connectors nor BoB category. The BIF categorizes three solutions in the category hybrid connectors. The first scheme involves *trusted relays* and typically appears in a permissioned blockchain as a trusted escrow party redirects transactions from a source blockchain to a target blockchain.

Blockchain-agnostic protocols are essential for BoB (discussed in the next section 2.3). They enable cross-chain communication between arbitrarily distributed ledger technologies by providing a blockchain abstraction layer that exposes a set of uniform operations allowing a decentralized application to interact with blockchains without the need of using different APIs.

Finally, *blockchain migrators* allow blockchain-state migration across blockchains. First, the state is locked on the source blockchain; then the state is recreated in the target blockchain.

2.3 Blockchain of Blockchains

The BIF identifies a third category of interoperability solutions, BoBs. Unlike sidechains, they are not an extension of one blockchain but rather independent blockchains that interoperate among each other. According to Belchior et al., the most widely adopted BoBs are Cosmos [23] and Polkadot [43]. This section discusses both, each with its implemented approach. The discussion is driven by the chosen security model, consensus, and cross-chain communication: first Cosmos, with the *network of heterogeneous state machines* approach (section 2.3.1), followed by Polkadot, with the *unified state machine with heterogeneous shards* approach (section 2.3.2).

2.3.1 Network of Heterogeneous State Machines

This is a decentralized network of independent blockchains (the *heterogeneous state machines*). The best known implementation of this approach is Cosmos. Independent blockchains called *zones* are connected by Cosmos network hubs, the digital ledgers responsible for data transfer between zones. They can transfer data to other zones directly or via hubs. Both methods utilize the Interblockchain Communication Protocol (IBC) protocol [14] by Goes.

Security Model

Cosmos uses a bridge-hub model to connect blockchains with independent security guarantees. As a result, inter-chain communication is bound by the trust that the source zone has in the target zone. Each zone is responsible for its security and must provide its block finalization by its own means.

Consensus

Cosmos uses the Tendermint [22] consensus protocol, a round-robin protocol with the provision of instant finality. That means it involves the production and finalization of one block at a time.

Cross-Chain Communication

Cosmos implements the IBC protocol for cross-chain communication. Currently, the protocol only supports the cross-chain transfer of tokens, which are blockchain-based abstractions of assets such as currency, resources, access, identity, or collectibles that a blockchain account can own [2]. Because each zones provides its own security, interacting zones must trust each other and accept the risks of each other's failures during cross-chain communication.

2.3.2 Unified State Machine with Heterogeneous Shards

Wood[1] has proposed Polkadot, an implementation of the unified state machine with heterogeneous shards approach. In this approach, the network consists of parallel application-specific blockchains called parachains that are connected to the main chain, known as the relay chain. Each parachain is a heterogeneous shard that can run in parallel to the relay chain.

Security Model

Polkadot has a shared security model. The relay chain contains the global state of the entire system and approves the state transition correctness of all connected parachains. This allows blockchain technologies that do not trust each other to interact with each other.

[1]Gavin James Wood, British computer scientist, is one of Ethereum's co-founders. While awaiting the release of a new Ethereum specification with a scalability solution, later known as sharding, Wood devised the vision for a heterogeneous multi-chain.

Consensus

Polkadot uses a hybrid consensus model with two algorithms. Blind Assignment of Blockchain Extension (BABE) [1] for block production and GHOST-based Recursive Ancestor Deriving Prefix Agreement (GRANDPA) [34] for block finalization. GRANDPA is a Byzantine Fault Tolerance (BFT) consensus protocol finalizing one of the existing chain of blocks without producing new blocks. The authors based their research on the idea that block finalizing means simultaneously finalizing of all its previous blocks. Therefore, there is no need to vote for each block but only for chains that may consist of hundreds or thousands of blocks. GRANDPA allows the reduction of the network load and an increase in system performance by approving a large number of blocks simultaneously. However, this may increase the delay of a particular block approval. Notably, GRANDPA and Tendermint are both BFT-based algorithms and showcase quadratic transport complexity. Cosmos, however, can only finalize one block at a time.

Cross-Chain Communication

Parachains communicate through the Cross-Chain Message Passing (XCMP) protocol, a queuing communication mechanism based on a Merkle tree. XCMP is designed to communicate arbitrary messages between parachains. Messages are sent together with the next parachain block (short: parablock), while the relay chain blocks include only the proof of post. All messages must be processed in a proper order, for which a chain of Merkle proofs is used. However, XCMP is still under development. Therefore, the stop-gap protocol is Horizontal Relay-routed Message Passing (HRMP). As soon as XCMP is fully developed, it can replace HRMP. The primary difference between the two is the data stored on the relay chain. In HRMP, the relay chain stores the full message with its payload. XCMP, on the other hand, will only store a reference to the payload. The target parachain will be responsible for decoding the message payload.

2.4 Summary

This chapter discussed blockchain-enabled supply chain approaches including the solo-chain approach with the smart contract and software adapter variant, as well as the multi-chain approach. A new approach is developed in the following chapters that builds on the insights from the multi-chain approach. This prototype should provide blockchain interoperability and circumvent transaction fees.

For an overview of the state of the art in blockchain interoperability solutions for the system to develop, the BIF was briefly introduced. It categorizes current solutions into the categories of public connectors, hybrid connectors, and BoBs. The BoB category was identified as the most suitable for the multi-chain approach. Therefore, the two widely used solutions in this category, Polkadot and Cosmos, were further discussed regarding their underlying approach, as well as security model, consensus, and cross-chain communication abilities.

Polkadot is an implementation of the unified state machine with heterogeneous shards approach. It uses a hybrid consensus model, separating block production (BABE) from finality (GRANDPA). This allows for blocks to be rapidly produced and finalized at a slower pace without risking slower transaction speeds or stalling. Unlike Cosmos, Polkadot provides a shared security model, cross-chain transfers of arbitrary data. The finalization of multiple blocks at a time justifies the quadratic transport complexity, thus providing the best foundation for the design and implementation of ChainFresh.

Part II

ChainFresh

Chapter 3

The ChainFresh Approach

The ChainFresh approach provides an implementation of the multi-chain approach (section 2.1.2). The blockchain consortium comprises a multi-chain ecosystem for organizations. Each organization is allowed to participate in the consensus process. The first section (3.1) of this chapter describes the solution approach in a nontechnical way. In the following section (3.2), an overview of the ChainFresh architecture is presented by briefly introducing each individual tier. Finally, the choice of substrate as the development framework for the blockchains of ChainFresh is justified in section 3.3.

3.1 The Benefits for Supply Chain Participants

This section highlights the two main benefits of the ChainFresh approach for supply chain participants: interoperability (section 3.1.1) and controlled transparency (section 3.1.2). For this purpose, the benefits are illustrated in the context of the following example, depicted in figure 3.1, which considers a single supply chain within an FSCN for organic strawberries with bio-farmer, processor, distributor, and retailer.

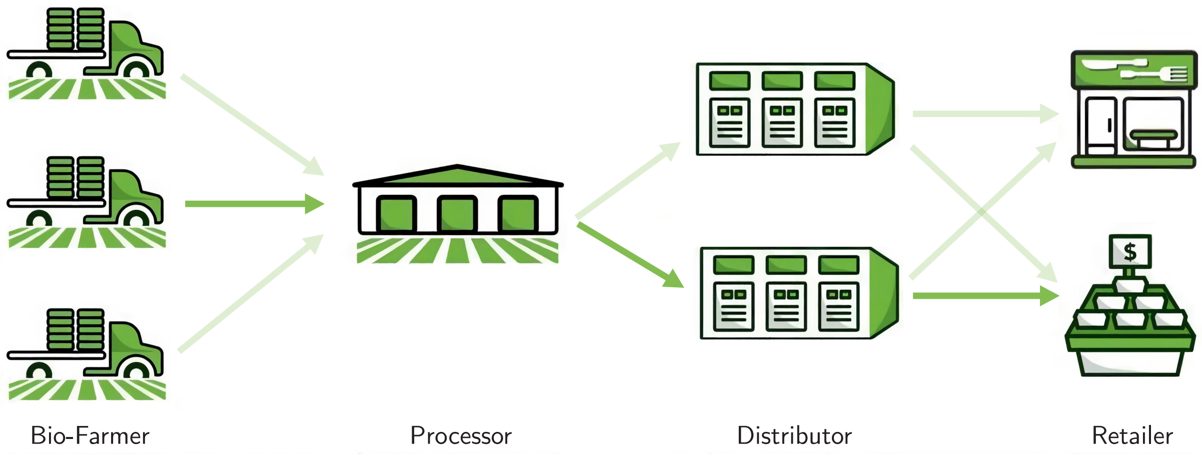


Figure 3.1: Illustrative example of a single supply chain (opaque) within an FSCN

3.1.1 Interoperability

According to Pan, Zhong, and Qu, interoperability should be considered at a physical level (e.g., standardized handling), organizational level (e.g., interorganizational protocols), business level (e.g., business models with shared value), and digital level (e.g., interoperable information systems) [27]. This thesis considers digital-level interoperability in the context of blockchain technology. The state of the art (chapter 3) initially identifies three types of blockchain interoperability: between different (homogeneous or heterogeneous) blockchains, between decentralized applications using the same blockchain (solo-chain), and blockchain in combination with other technologies (e.g., enterprise systems).

ChainFresh aims to provide a permanent and shared record of food system data that connects participants across the FSCN. This is done through the use of a main blockchain, the ChainFresh relay chain. The sole purpose of the relay chain is to coordinate and share appropriate data and ensure all participants are complying to overarching rules of the consortium. Supply chain participants can set up their own ChainFresh parachain, which is a blockchain that runs in parallel to the ChainFresh relay chain. A parachain has a set of predefined modules for managing access, documents, products and shipments and enabling interoperability. Therefore, ChainFresh provides interoperability among different heterogeneous blockchains. Supply chain participants can add or remove their parachain from the consortium. For instance, a distributor can join the consortium blockchain of a food supply chain when delivering shipments from a processor to a supermarket. After the shipments have been delivered, the distributor can remove their parachain. Because the relay chain maintains a permanent and shared record of food system data, the information will not be lost for the consortium after the distributor has left. Through interoperability, organizations can share immutable data with other participants in their supply network.

3.1.2 Controlled Transparency

Before defining controlled transparency and how ChainFresh aims to provide it, the term transparency must be defined in the context of supply chains. Egels-Zandén, Hulthén, and Wulff [9] have articulated that transparency "comprises corporate disclosure of (i) the names of the suppliers involved in producing the firm's products (i.e., traceability), (ii) information about sustainability conditions at these suppliers, and (iii) the buying firms' purchasing practices" (ch. 2, p. 96).

In the context of ChainFresh, controlled transparency allows each organization to set up and manage its own parachain. This allows an organization to pinpoint the communication with trading partners by opening messaging channels only when they are required for transferring sensitive information, which should not be accessible by other organizations. For instance, a processor can choose to share a delivery document to only one of his distributors. Every other participant should not be able to see the content of the document. This does not effect the overall transparency, because information about product conditions and the parties involved in delivering the food from bio-farmer to retailer are openly shared among the supply chain participants.

3.2 Architectural Overview of ChainFresh

ChainFresh as a distributed system is a composition of three tiers: presentation, application, and relay. Each tier is composed of multiple components defining its distinct scope of work. A high-level architecture overview is outlined in figure 3.2.

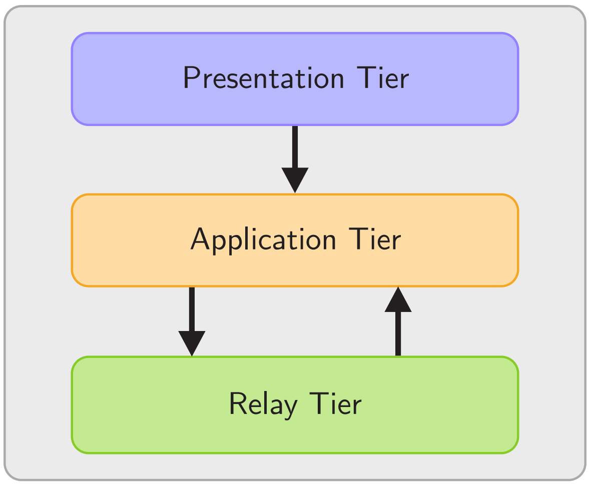


Figure 3.2: The three-tier system architecture of ChainFresh

3.2.1 Presentation Tier

To provide the user with convenient access to the ChainFresh system, the presentation tier is responsible for interacting with the application tier through a websocket connection. Any websocket-capable client or device can communicate with the endpoints exposed by the application tier. The user interacts with a GUI to manage the permissions of participating members, register shipments and products, and trace shipments along the supply chain. A browser extension is required to manage blockchain accounts and to sign transactions within those accounts.

3.2.2 Application Tier

The application tier consists of the parachains, their runtime modules, and off-chain processes. Collators are the actors of the parachains. Collators are responsible for collating user transactions and producing state-transition proofs for the validators of the parachain to find consensus. Collators do not directly participate in the consensus process. The primary modules of a collator are the business logic, Cumulus, Off-Chain Worker (OCW), and transaction pool. Figure 3.3 illustrates the application tier and how the collator node modules interact to produce a parablock. Application-specific business logic is executed based on the extrinsics (transactions) in the transaction pool (holds all valid and invalid transactions) and realizes the main functionality within the ChainFresh system: access control using Role-based Access Control (RBAC); managing decentralized identities for member organizations; registering products, shipments, or documents; and monitoring shipments through the supply chain.

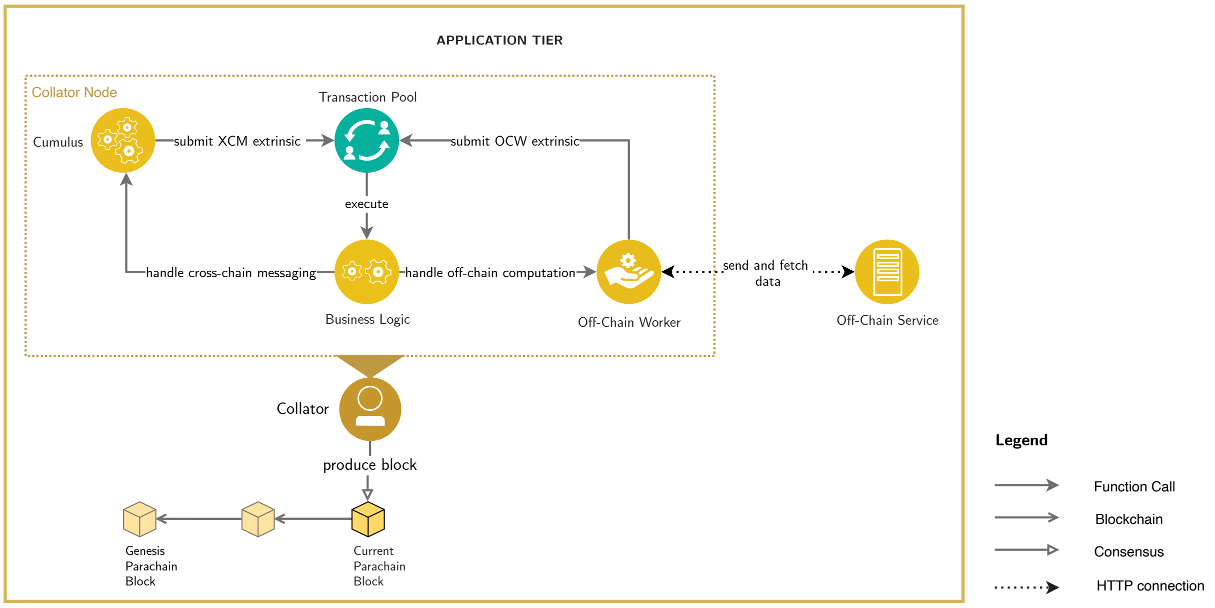


Figure 3.3: The application tier of the ChainFresh system

The business logic is decomposed in tightly coupled modules called pallets that conform to the Framework for Runtime Aggregation of Modularized Entities (FRAME). The business logic makes use of Cumulus for handling Cross-Chain Messaging (XCM) and the OCW for off-chain tasks. The OCW is primarily used for sending and fetching data to and from the off-chain service via HTTP for product-tracking status updates. Once Cumulus and the OCW have finished their task, they submit an extrinsic to the transaction pool for inclusion in a block. The business logic itself is decomposed into tightly coupled modules called pallets that conform to the FRAME. Figure 3.4 depicts the business logic pallets, each with its provided functionality that can be invoked with extrinsics.

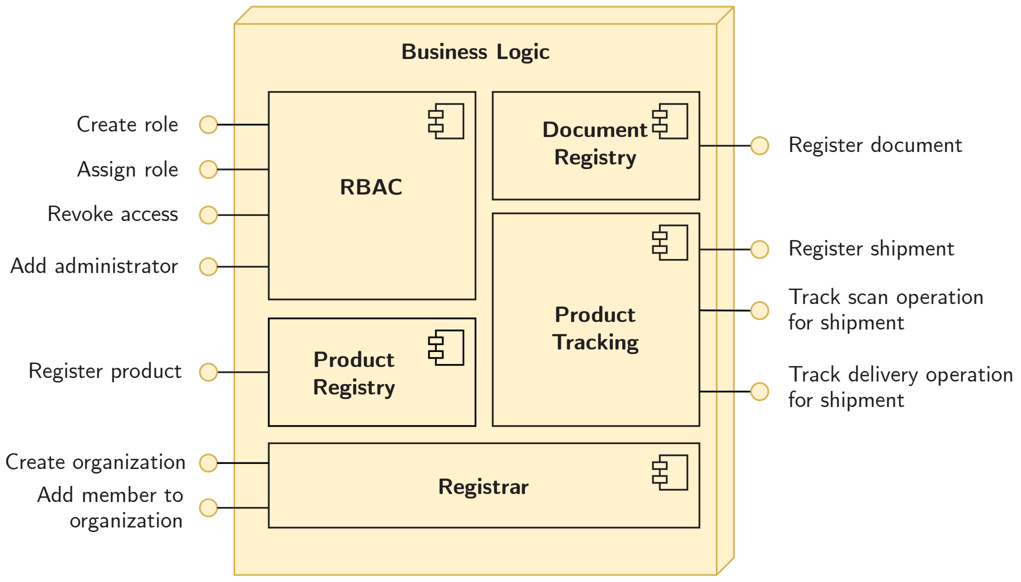


Figure 3.4: An overview of the business logic, decomposed into pallets

3.2.3 Relay Tier

The relay chain, in the relay tier, is the central hub in the network of heterogeneous blockchains, the parachains. The relay chain, depicted in figure 3.5, provides parachains with parablock validation and allows them to communicate with each other using the XCM format for cross-chain messaging.

Validators are the actors of the relay chain and have three responsibilities: (1) to verify that the information contained in parablocks is valid such as the identities of the transacting parties, (2) to participate in the consensus mechanism to produce the relay chain blocks based on validity statements from other validators, and (3) to handle cross-chain messages. For validators to fulfill their responsibilities, they are equipped with six primary runtime modules. The *inclusion* module handles the inclusion and availability of parablocks. In addition, *shared* manages the shared storage and configurations for other validator modules. The *paras* module manages the chain-head and validation code for parachains. The *scheduler* is responsible for parachain scheduling as well as validator assignments for the consensus mechanism. The *validity* module addresses secondary checks and disputes resolution for available parablocks. Finally, the *XCMP* module handles cross-chain messages and ensures that the messages are relayed to the receiving parachain.

An integral part of cross-chain communication is the establishment of a cross-chain messaging channel between the validators of two communicating parachains. Burdges et al. [7] have stated that a messaging channel aims to guarantee four things: “First that messages arrive quickly; second that messages from one parachain arrive to another in order; third that arriving messages were indeed sent in the finalised history of the sending chain; and fourth that recipients will receive messages fairly across senders, helping guarantee that senders never wait indefinitely for their messages to be seen” (section 4.4.3, p. 19).

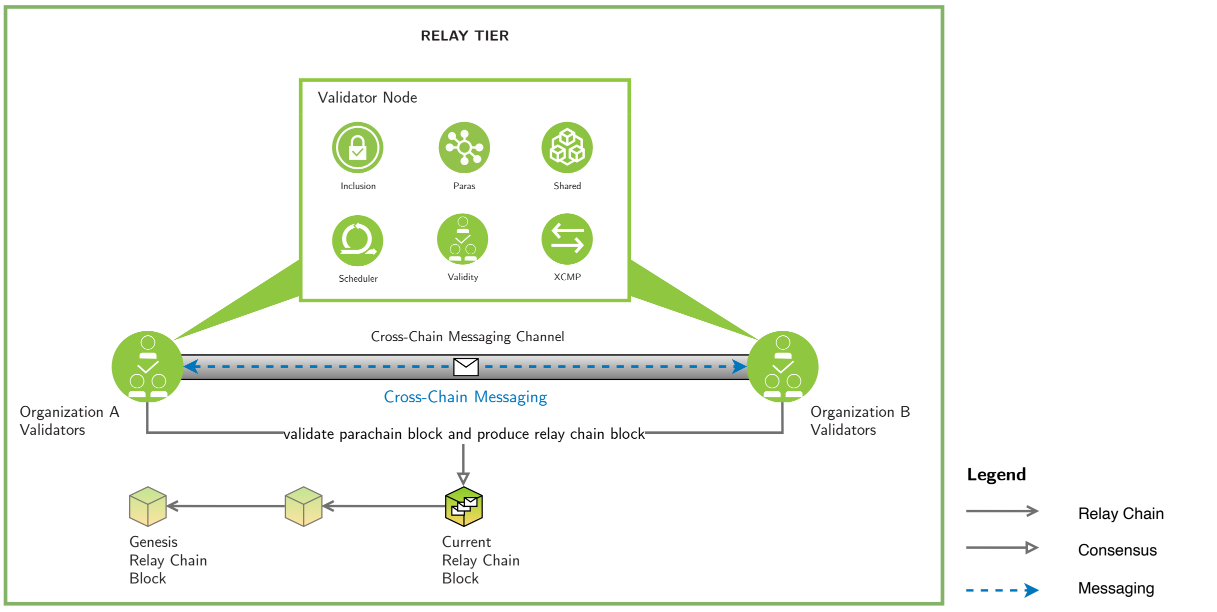

RELAY TIER
Validator Node
Inclusion
Paras
Shared
Scheduler
Validity
XCMP
Cross-Chain Messaging Channel
Cross-Chain Messaging
Organization A Validators
Organization B Validators
validate parachain block and produce relay chain block
Genesis Relay Chain Block
Current Relay Chain Block
Legend
Relay Chain
Consensus
Messaging

Figure 3.5: The relay tier of the ChainFresh system

3.3 Decision for the Substrate Framework

ChainFresh is built with substrate[35], a modular framework for building blockchains. A nontechnical reason for using substrate is its flexibility. Organizations must be able to adapt their blockchain system to meet supply chain compliance requirements of regulatory bodies. Regulations happen frequently, especially in food supply chains, as shown in section 1.1.2. Due to the modular nature of substrate-based blockchains, developers have the necessary freedom to swap or add modules to their blockchain runtime.

Technical reasons include the chosen programming language, the software design, and the off-chain abilities. Substrate is implemented in the programming language Rust, which aims to provide performance (comparable to C++), reliability, and better means for productivity. In terms of reliability, Rust manages resources (including memory, files, network, and thread) and avoids problems such as resource leaks or data races. Finally, for productivity, Rust provides Integrated Development Environment (IDE) support and type inspections. Furthermore, substrate is generic by design, meaning transactions are abstracted to extrinsics (things that happen outside the chain) and intrinsics (things that happen inside the chain). Transactions are stored as binary large objects. As a result, users can transfer and store any type of data on the blockchain.

Nonetheless, with ChainFresh as a permissioned blockchain, concerns about off-chain processes need to be raised. For instance, Helliar et al. have posited that "off-chain processes may become a major barrier for permissioned blockchains" [18]. Using substrate, off-chain data can be queried or processed before it is included in the on-chain state through OCW, a collator node subsystem that allows for the execution of long-running and possibly nondeterministic tasks. Moreover, an OCW does not influence the block production time.

3.4 Summary

ChainFresh implements the multi-chain approach for supply chains from section 2.1.2. The ChainFresh system is split into three tiers: presentation, application, and relay (refer to appendix A for a combined overview of the three tiers presented in this chapter). Relay chain validator nodes and parachain collator nodes are architecturally different and use different modules to fulfill their responsibilities. The collators produce parablocks and rely on the validators to validate the transactions inside these blocks.

In summary, the architecture of ChainFresh adheres to the following design principles:

1. Consortium: Organizations are allowed to join the network and participate in the consensus processes of the relay chain.
2. Heterogeneous: The parachains are sovereign over their state transitions. Thus, organizations are able to extend the runtime of their parachains with their application-specific logic, encapsulated in FRAME pallets.

3. Interoperability: The parachains use the relay chain consensus and validation for cross-chain communication and shared security.
4. Controlled transparency: Organizations can store immutable data on their own parachain. As a result, organizations have full control over exposure of data to, and acceptance of data from, other parachains.

Chapter 4

Presentation Tier

The presentation tier encompasses the web frontend and a Web3 browser extension. The web frontend is built with NodeJS and uses the Polkadot JavaScript client library to interact and query the application tier nodes. The capabilities that the client library expose are implemented on top of the substrate Remote Procedure Call (RPC) API. The GUI of the web frontend is the dashboard. Figure 4.1 depicts the main view of the dashboard and its contents. Located at the top of the dashboard is the navigation menu with the following menu items. The “Dashboard” item (1) allows a user to return to the start view. The “Explorer” item (2) redirects a user to the blockchain explorer that is part of the hosted Polkadot{.js} Apps[1] service. Appendix A depicts a screenshot of the blockchain explorer, which shows the most recent blocks as they become available. The explorer connects to a local websocket endpoint to retrieve the blockchain data.

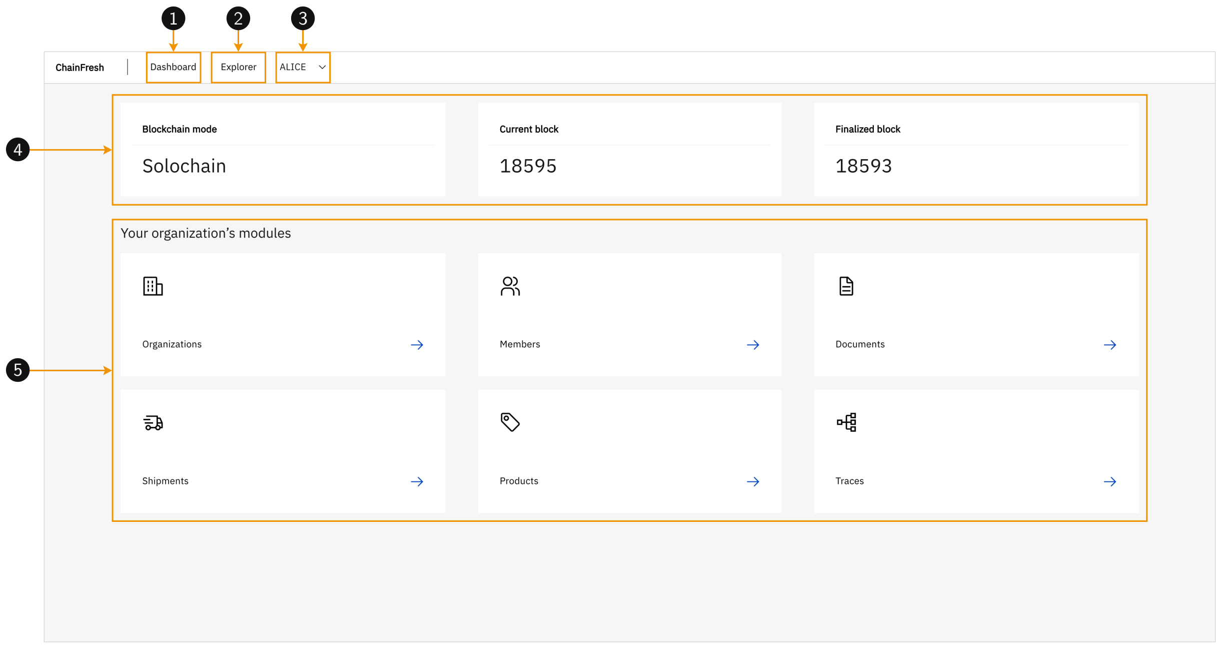


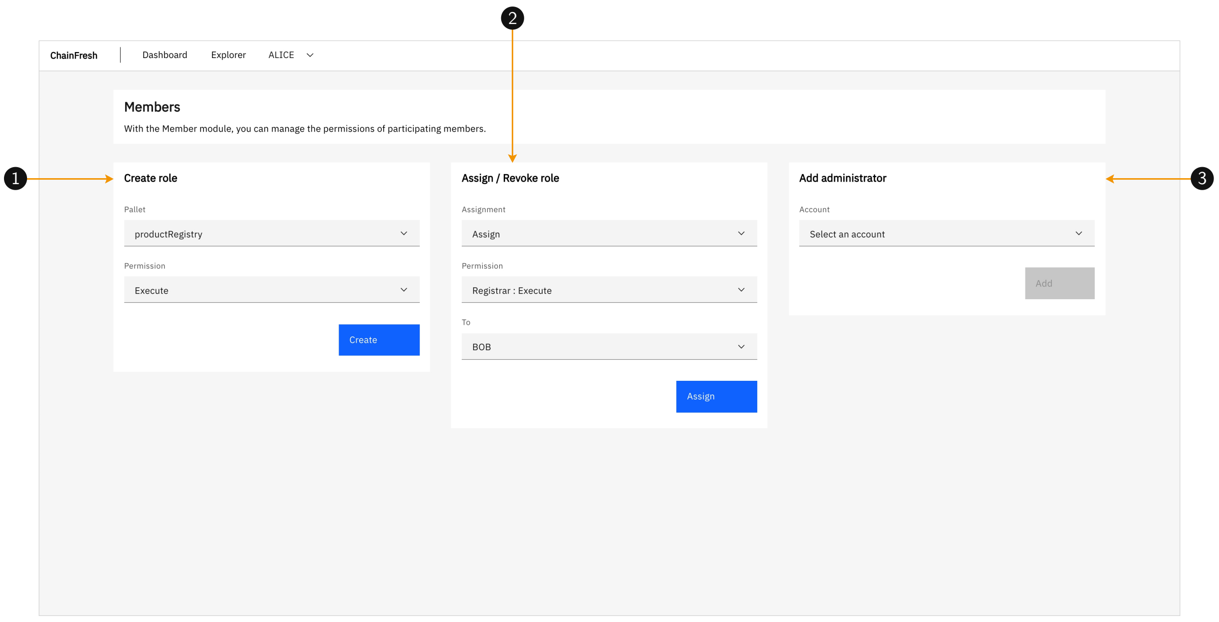
Figure 4.1: The dashboard main view

[1]https://polkadot.js.org/apps/#/explorer.

The account selector dropdown (3) allows users to choose an account for performing actions. User accounts are employed in a variety of contexts in ChainFresh, including as identifiers and for signing extrinsics. To request a signature from the user or have the user approve a transaction, one must be able to access the user's accounts. User accounts are managed by a Web3 browser extension. It enables performing operations on a set of keys (e.g., sign, verify) without exposing the secret key to the outside world. Below the navigation menu is a common area for displaying the view content. The content area of the dashboard main view provides information about the blockchain mode, current block, and finalized block of the connected blockchain (4). The dashboard is divided into different views: organizations, members, documents, products, shipments, and traces. Each view can be accessed via the view selection (5).

4.1 "Members" View

The "members" view allows an administrator (a user with the `Rbac:Manage` role) to create roles, assign or revoke roles, and add administrators to the organization. To create a role, the "create role" form (1) requires two selections: the pallet and the permission (execute or manage). Next is (2) assign or revoke role: Assignment Assign/Revoke. Here, the permission shows the combination `Pallet:Permission` of the previously created role Finally, an administrator of an organization can add another administrator by selecting the other's account (3). Figure 4.2 shows the organization administrator, Alice, creating a role for the pallet `productRegistry` with execute permission. Alice has manage permissions, to allow her to join the blockchain consortium and bootstrap the parachain for her organization. Additionally, she assigns a previously created role `Registrar:Execute` to Bob, allowing him to submit extrinsics to the organizations registrar.



Members

With the Member module, you can manage the permissions of participating members.

Create role

Pallet: productRegistry

Permission: Execute

Assign / Revoke role

Assignment: Assign

Permission: Registrar : Execute

To: BOB

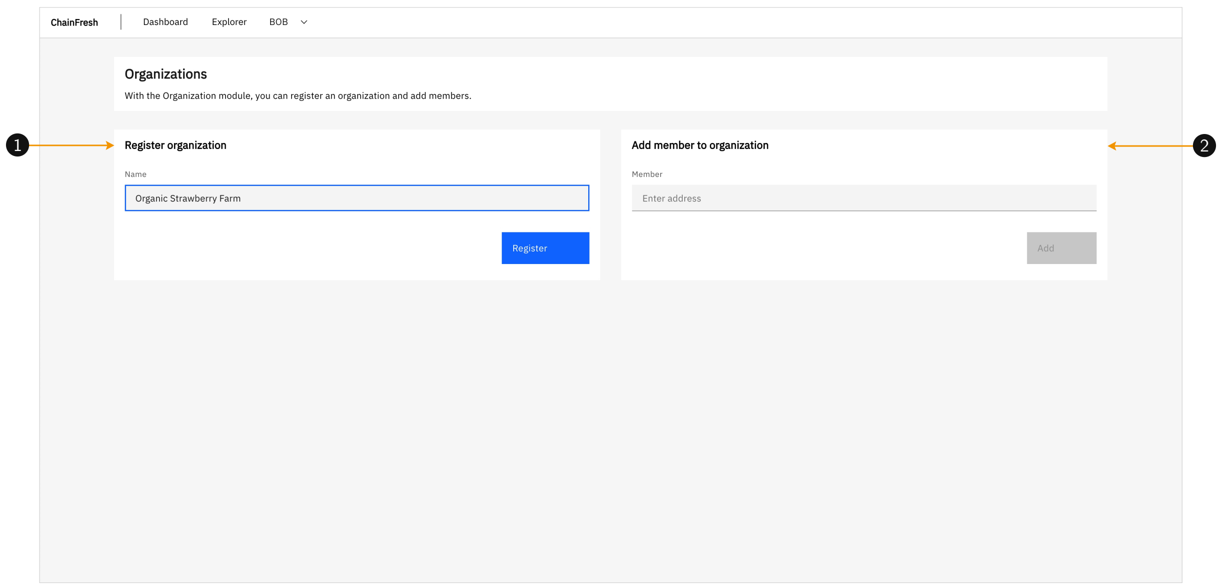
Add administrator

Account: Select an account

Figure 4.2: The members view

4.2 "Organizations" View

The view "organizations" (figure 4.3) provides a user with the ability to register an organization, or to add a member to a registered organization. Using the text input field in "create organization" (1), a user can enter the name of the organization to register in the ChainFresh system. An organization can only be registered once and by one user. Upon registration, the user is referred to as the organization owner. An organization owner is the only account that may add members to its organization by entering the corresponding blockchain address in "add member to organization" (2). The added user account becomes a delegate of the organization. In figure 4.3, the user Bob registers an organization named Organic Strawberry Farm. This allows him in the following sections to register documents, products, and shipments under the name of the organization.



Organizations

With the Organization module, you can register an organization and add members.

Register organization

Name: Organic Strawberry Farm

Add member to organization

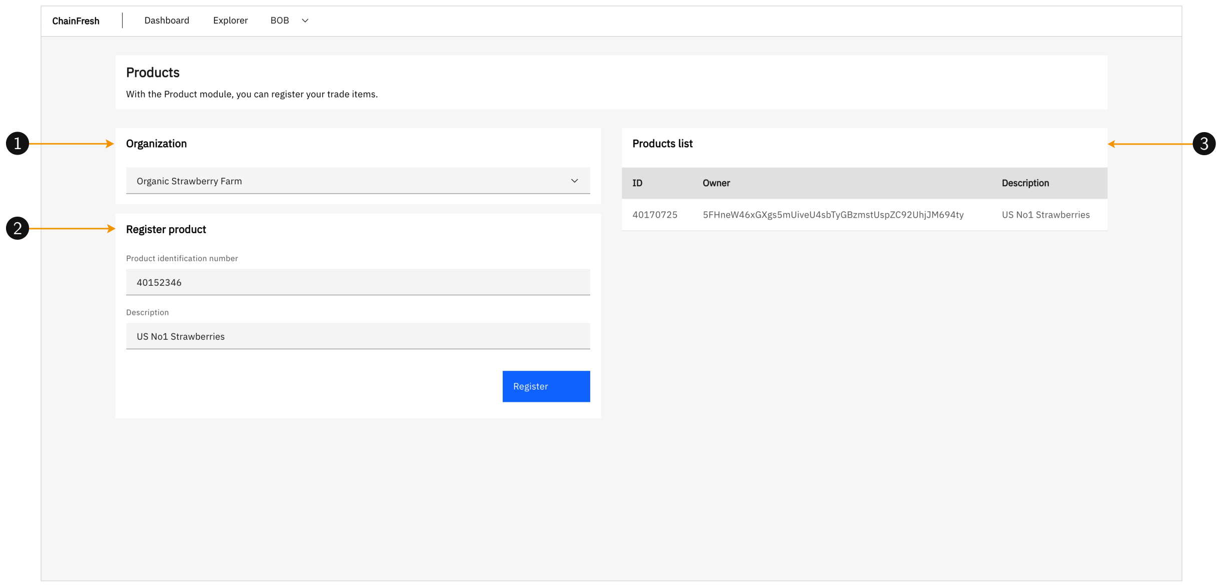
Member: Enter address

Figure 4.3: The organizations view

4.3 "Products" View

The view "products" allows a user to register products in the ChainFresh system. Located on the left side of the view is an organization selector (1), allowing a user who is part of multiple organizations to switch between them, and the "register product" form (2) on the left side of the view has two text input fields. The product identification number typically is a numeric or alpha-numeric code such as a GS1[2] Global Trade Item Number (GTIN) [16]. Custom codes are possible as well as long as they follow a well-defined data structure, like GTIN. The description text input field is for additional information about the product content. All registered products, each with its id, owner, and description, are listed in the "product list" (3) on the right side of the view. A newly registered product appears in the list as soon as it is included in a parablock.

[2]GS1 standards enable products and services to be identified by uniquely identifiable barcodes.



Products

With the Product module, you can register your trade items.

Organization

Organization Strawberry Farm

Register product

Product identification number

40152346

Description

US No1 Strawberries

Products list

ID	Owner	Description
40170725	5FHneW46xGXgs5mUiveU4sbTyGBzmstUspZC92UhjJM694ty	US No1 Strawberries

Figure 4.4: The products view

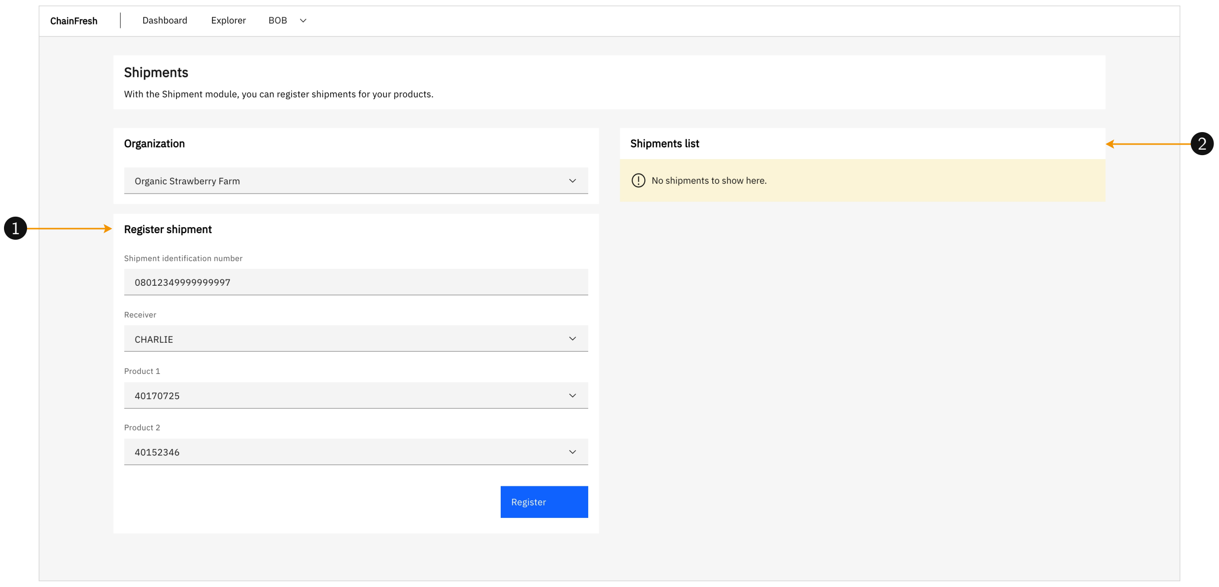
In figure 4.4, user Bob registers a product under the organization Organic Strawberry Farm. He enters a product identification number in GTIN-8 format and describes the product content using USDA grades and standards for strawberries [38]. US No. 1 marks them as high-quality strawberries.

4.4 “Shipments” View

To register a shipment, one can use the “register shipment” form (1), which requires at least three inputs: shipment identification number, receiver, and one product. Optionally, a second product can be selected.

The shipment identification number, like the product identification number, follows the GS1 standard for Global Shipment Identification Number (GSIN) [15]. The receiver is the organization owner account of the organization that receives the shipment. All registered shipments of the organization, each with its id, owner, receiver, and associated products, are listed in the “shipments list” (2).

In figure 4.5, Bob is registering the first shipment; thus, a message is displayed instead of the shipments list. To register the shipment, Bob enters a shipment identification number in the GSIN standard. Next, he selects the receiver, Charlie. Finally, he selects the two products from the previous section (4.3) to be included in the shipment. Because no shipment has been registered yet, Bob sees a clue that indicates the circumstance instead of seeing a list of shipments.



Shipments

With the Shipment module, you can register shipments for your products.

Organization

Organic Strawberry Farm

Register shipment

Shipment identification number

08012349999999997

Receiver

CHARLIE

Product 1

40170725

Product 2

40152346

Shipments list

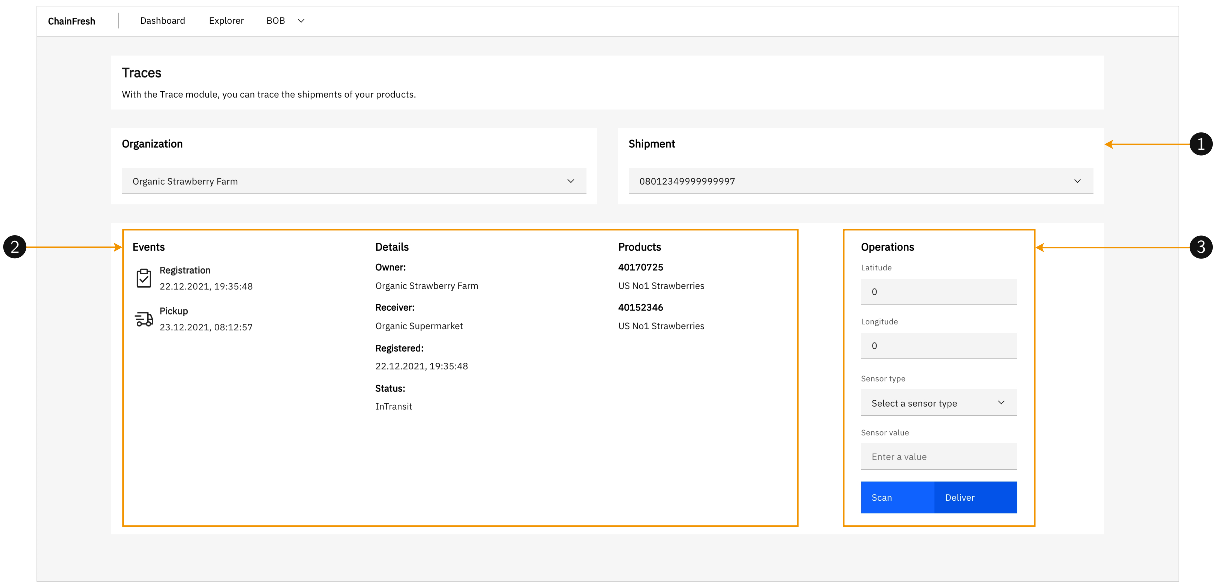
No shipments to show here.

Figure 4.5: The shipments view

4.5 "Traces" View

With this view, shipments can be traced along the supply chain. The "shipment" selector (1) allows one to select a shipment for tracing. Once a shipment is selected, an additional section appears with shipment details and operations. The shipment details (2) are split into three columns: events, details, and products. Events reflect the operations associated with the selected shipment that previously occurred. Details provide information about the shipment owner and receiver, when the shipment was registered, and the current status of the selected shipment. The status of a shipment can be "pending," "in transit," or "delivered." Products, as the name suggests, shows the products contained in the shipment. The "operations" column (3) allows one to enter values for latitude, longitude, sensor type, and sensor value and has two buttons. The input values are intended for a scan operation, which can be recorded with the "scan" button. A delivery of a shipment can be recorded using the "deliver" button.

Figure 4.6 depicts user Bob tracing a shipment to Charlie's Organic Supermarket from the previous section. The shipment details reveal that the shipment is currently in transit and that is contains the two previously registered products.



Traces

With the Trace module, you can trace the shipments of your products.

Organization: Organic Strawberry Farm

Shipment: 08012349999999997

Events

Registration
22.12.2021, 19:35:48

Pickup
23.12.2021, 08:12:57

Details

Owner:
Organic Strawberry Farm

Receiver:
Organic Supermarket

Registered:
22.12.2021, 19:35:48

Status:
InTransit

Products

40170725
US No1 Strawberries

40152346
US No1 Strawberries

Operations

Latitude: 0

Longitude: 0

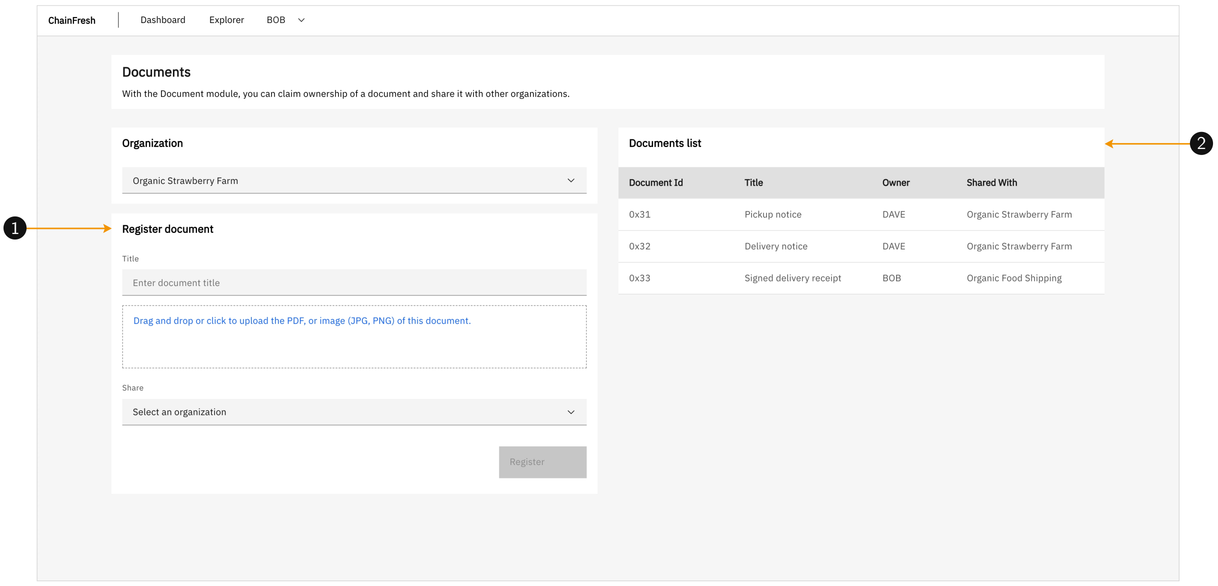
Sensor type: Select a sensor type

Sensor value: Enter a value

Figure 4.6: The traces view with a selected shipment

4.6 "Documents" View

This view allows supply chain participants to share documents with each other. Using the "register document" form (1), a user can register a document by entering the document title, drag and drop or click to upload the PDF or image (JPG, PNG) for the corresponding document, and optionally choose with whom to share the document. A user can choose to share a document only with their own organization or with another organization in the system. When a document is registered in the system and shared with the organization of the current user, it is displayed in the document list (2). Figure 4.7 illustrates the exchange of documents.



Documents

With the Document module, you can claim ownership of a document and share it with other organizations.

Organization: Organic Strawberry Farm

Register document

Title: Enter document title

Drag and drop or click to upload the PDF, or image (JPG, PNG) of this document.

Share: Select an organization

Documents list

Document Id	Title	Owner	Shared With
0x31	Pickup notice	DAVE	Organic Strawberry Farm
0x32	Delivery notice	DAVE	Organic Strawberry Farm
0x33	Signed delivery receipt	BOB	Organic Food Shipping

Figure 4.7: The documents view

Chapter 5

Application Tier

This chapter presents the application tier in detail. First, the collator node architecture of a parachain is described in section 5.1. The collator node runtime is composed of five FRAME pallets that define its business logic: RBAC (section 5.2.1), registrar (section 5.2.2), document registry (section 5.2.3), product registry (section 5.2.4), and the product-tracking pallet for tracing shipments along the supply chain (section 5.2.5).

5.1 Collator Node Architecture

A collator node is composed of six components: storage, runtime, P2P networking, RPC, consensus, and telemetry. Figure 5.1 illustrates the architecture of a substrate-based collator node. Each component is briefly explained in the following.

The blockchain network allows participants to reach trustless consensus about the state of storage. The *storage* component is responsible for the evolving state of a substrate-based blockchain. To achieve this, substrate implements RocksDB [30], a key-value storage mechanism for fast storage environments.

Runtime defines the state-transition logic. It is compiled to WebAssembly (Wasm) and made part of the blockchain's storage state. A node may include a native runtime compiled for the same platform as the node itself. The executor, responsible for dispatching calls to the runtime, selects between native code and interpreted Wasm. While a native runtime can offer performance advantages, the executor will choose to interpret the Wasm runtime if it implements a newer runtime version.

The *P2P networking* component provides necessary capabilities, such as P2P routing, for nodes to communicate with each other. Substrate uses the Rust implementation of the LibP2P network stack [24].

Substrate provides HTTP and WebSocket RPC servers, encapsulated in the *RPC* component. The RPC capabilities allow blockchain users to interact with the node and the network.

The *consensus* component contains the logic that allows network participants to agree on the state of the blockchain. Substrate ships with several consensus mechanisms (e.g., proof of work). One can also supply a custom consensus engine.

The *telemetry* component exposes client metrics via the embedded Prometheus server (e.g., LibP2P peers count). Node operators can use the metrics for Prometheus-/Grafana-based monitoring systems.

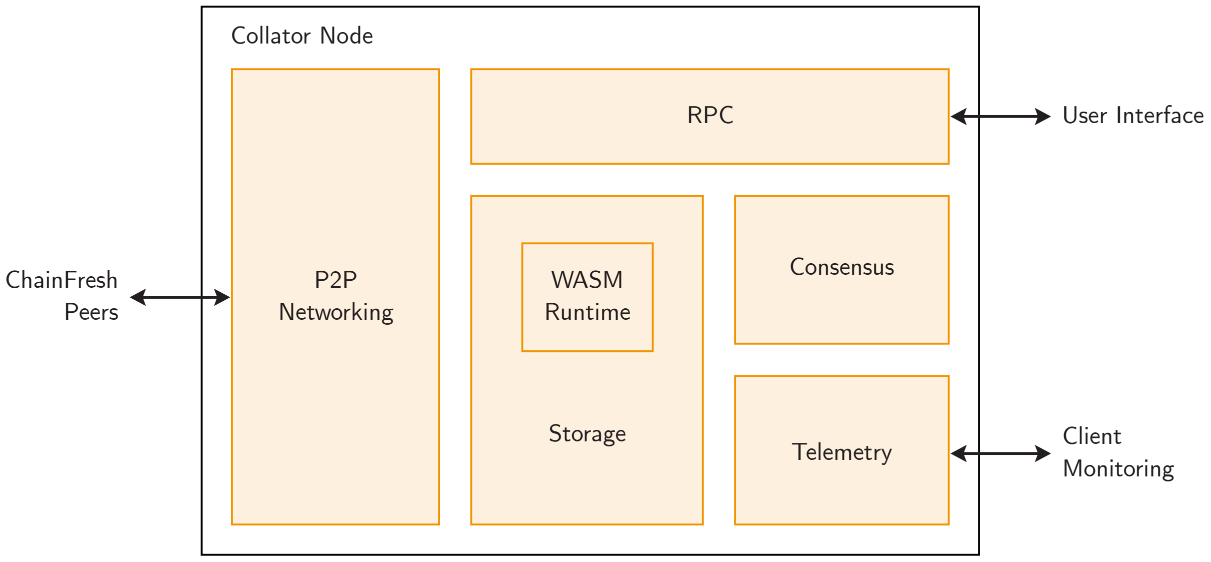


Figure 5.1: Substrate node architecture (adopted from Parity [28])

5.2 Business Logic

The business logic is encapsulated in pallets. The general seven-section structure of a pallet is as follows:

1. *Imports and Dependencies*: Imports of modules that the pallet depends on.
2. *Pallet Type Declaration*: A placeholder to implement Rust traits and methods.
3. *Runtime Configuration Trait*: All types and constants for the runtime to use.
4. *Runtime Storage*: Declaration of runtime storage items that are persist between blocks and can be accessed by the runtime logic.
5. *Runtime Events*: Definition of the pallet's events. Events can be emitted to notify external entities about changes or conditions in the runtime.
6. *Hooks*: The pallet logic that should be executed regularly in some context (for instance, `on_initialize`, which is called before all extrinsics).
7. *Extrinsics*: Dispatchable functions allow users to interact with the pallet and invoke state changes from outside the runtime. These functions materialize as "extrinsics," which are often compared to transactions.

Extrinsics do not require transaction fees. Instead, the dispatchable functions are annotated with transaction weights. A transaction weight is a representation of block execution time (i.e., weight is another representation of time). In the ChainFresh implementation, the block execution time is six seconds. Transaction weight is defined as follows:

$$1 \text{ second} = 10^{12} \text{ weight}$$
$$1000 \text{ weight} = 10^{-9} \text{ seconds}$$

Substrate makes several mechanisms available to manage access to resources and to prevent individual components of the chain from consuming too much of any resource.

5.2.1 Role-Based Access Control

To control the access in terms of who can submit extrinsics, ChainFresh implements RBAC formalized by Ferraiolo, Kuhn, and Chandramouli [11]. RBAC has become the predominant model for user access control.

The `rbac` pallet maintains an on-chain registry of roles and the users to which those roles are assigned. A role is a tuple with the name of a pallet and a permission that qualifies the level of access granted by the role. A permission is an enum with the variants Execute and Manage. The Execute permission allows a user to invoke a pallet's dispatchable functions. The Manage permission allows a user to assign and revoke roles for a pallet and also implies the Execute permission. Access control validation is done within the transaction pool validation by way of the RBAC pallet's Authorize signed extension. Therefore, permissions are configured in the chain specification file.

The `rbac` pallet exposes the following extrinsics:

- `createRole(pallet, permission)`: Create a role for a pallet with permission.
- `assignRole(address, role)`: Assign the role for a given address.
- `revokeAccess(address, role)`: Revoke the role for a given address.
- `addSuperAdmin(addressTo)`: Add an administrator.

5.2.2 Registrar

The registrar inherits Decentralized Identifier (DID) [33] capabilities from the DID pallet and uses these capabilities to implement an organization registry. This pallet maintains a list of organizations and maps each organization to a list of members.

Organizations are identified by the account ID that registered it. They are also associated with a name, which is designated by the value of the `Org` attribute on the DID of the organization owner. Organization owners are the only accounts that may add members to their organizations. When an account is added to an organization as a member, the organization owner creates an `OrgMember` delegate for the member's DID. In this way, the organization owner can certify an account's membership in the organization. The registrar pallet exposes a custom origin, `EnsureOrg`, that validates whether or not an account owns or is a member of at least one organization. The `EnsureOrg` origin is used to control access to many of the chain's capabilities, including the ability to create roles with the RBAC pallet.

The `registrar` pallet exposes the following extrinsics:

- `createOrganization(orgName)`: Create an organization.
- `addToOrganization(addressTo)`: Add a member to the organization.

5.2.3 Document Registry

The document registry provides functionality for registering documents. A document can either be registered under the name of one organization or shared with other organizations in the supply chain.

The `documentRegistry` pallet uses proof of existence. Instead of storing the original document in the blockchain, only a hash value generated by the web frontend, a proof that the document exists, is stored. This requires (i) a user, (ii) a file hash (or file digest), and (iii) the timestamp of the verification. Before a document is registered, the digest is compared to all stored document digests to ensure that the same document can be registered at most once. The runtime storage is only modified after all checks are completed. This is important because if the transaction fails at some later point, the storage is modified and will remain so. The functionality for sharing documents is explained in section 6.3.

The `documentRegistry` provides one extrinsic `registerDocument` with the following parameters:

- `organization`: The organization account with which to share the document.
- `title`: The title of the document.
- `digest`: The file digest to store on the blockchain.

5.2.4 Product Registry

The product registry provides functionality for registering products (also known as trade items) exchanged in a supply chain among various stakeholders. A product owner (i.e., a member of an organization) registers product data in the system to be visible for other supply chain participants.

The `productRegistry` maintains a registry of products, which maps each product to the organization it belongs. The product registry provides one extrinsic `registerProduct` with the following parameters:

- `owner`: The account that represents the owner of this product, typically the manufacturer or supplier providing this product within the supply chain.
- `props`: A series of properties describing the product. Typically, there would at least be a textual description. It could also contain lot master data (e.g., expiration, weight, harvest date).
- `registered`: The timestamp at which the product was registered on the blockchain.

On successful registration, the `productRegistry` assigns a unique ID to the registered product for systemwide identification. The origin trait `EnsureOrg` is used to control the accounts that are allowed to create products.

5.2.5 Product Tracking

The `ProductTracking` pallet provides functionality for tracking shipments. One can monitor the storage and transportation conditions for a shipment along the supply chain.

Shipments are associated with an organization and have an assigned ID. This pallet supports tracking several types of shipping events: registration, pickup, scan, and delivery. With the exception of registration, shipment events may be associated with a list of sensor readings. The high-level flow is depicted in figure 5.2. The operations manager registers a shipment, and the `ShipmentStatus` turns to "pending." When the company transport operator tracks the pickup operation, the `ShipmentStatus` turns to "in transit." During the transport, multiple scan operations can occur, but the `ShipmentStatus` does not change. Finally, when a shipment arrives at its destination, the transport operator can track a delivery operation, and the `ShipmentStatus` turns to "delivered."

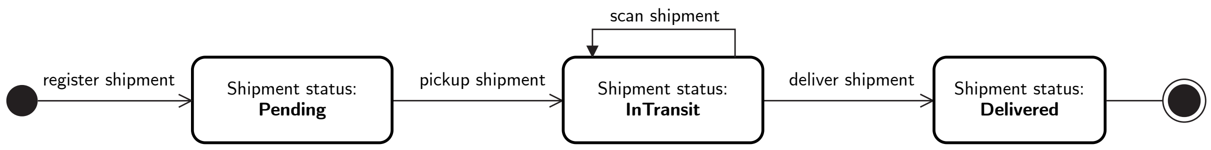


Figure 5.2: State graph of shipment updates

To construct a trace, `productTracking` implements the following two builders according to the builder pattern [13]: The `ShipmentBuilder<AccountId, Moment>` builds a traceable shipment.

`fn identified_by(mut self, id:ShipmentId)`: Requires a reference to a shipment that is already registered in the system.

`fn owned_by(mut self, owner:AccountId)`: The owner of the shipment.

`fn received_by(mut self, owner:AccountId)`: The receiver of the shipment.

`fn with_products(mut self, products:Vec<ProductId>)`: A series of products that are uniquely identified by their ProductId associated with the shipment.

`fn registered_at(mut self, registered:Moment)`: The moment at which the shipment was registered on-chain.

Next, the `ShippingEventBuilder<Moment>`

`fn at_location(mut self, location:Option<ReadPoint>)`: Location is an optional `ReadPoint` that contains the geographic position (latitude and longitude) where the event was captured.

`fn at_time(mut self, timestamp:Moment)`: Timestamp as time (represented as UNIX time) at which the event was captured by an external system or sensor.

`fn for_shipment(mut self, id:ShipmentId)`: Assignment of the shipping event to a tracked shipment based on the `ShipmentId`.

`fn of_type(mut self, event_typ:ShippingEventType)`: The business operation that occurred during the shipping process: pickup, scan, or deliver.

`fn with_readings(mut self, readings:Vec<Reading<Moment>>)`: Readings are an optional series of `Reading` that represent data captured by various sensors (humidity, temperature, vibration, etc). A `Reading` includes a `device_id` (unique identifier of the device), a `reading_type` (type of sensor/measurement), a `timestamp` (time at which the reading was recorded), and a value as the actual measurement recorded by the sensor.

To communicate with the outside world about the product tracking and conditions, the collator node architecture is extended with an OCW subsystem depicted in figure 5.3. Shipment events are placed in a queue that is monitored by an OCW. When events appear in this queue, the OCW sends them to an off-chain service via a REST API.

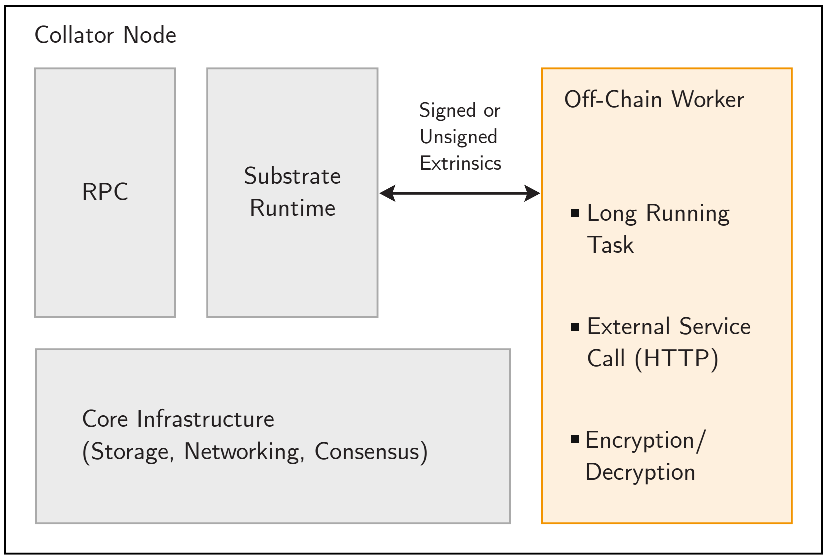

Collator Node
RPC
Substrate Runtime
Signed or Unsigned Extrinsics
Off-Chain Worker
Long Running Task
External Service Call (HTTP)
Encryption/ Decryption
Core Infrastructure (Storage, Networking, Consensus)

Figure 5.3: Collator node architecture with the OCW subsystem (adopted from Parity [28]). Core subsystems in gray, OCW subsystem in orange.

5.3 Cumulus

Parachains require a set of functionalities to send and receive messages and enable validators to validate their state transition for shared security. The Cumulus[1] subsystem provides interfaces and extensions to convert a substrate FRAME runtime into a parachain runtime. When compiling a parachain runtime, a Wasm binary with the parachain runtime code and the `validate_block` functionality is generated. This binary is required by the relay chain to register a parachain. The parachain consensus will follow the relay chain to get notified about which parablocks are included in the relay chain and which are finalized.

5.4 Summary

In summary, the application tier encompasses application-specific blockchains (the parachains) that allow organizations to join with their own blockchain, where they can store immutable data. Through this, organizations are able to create products and shipments. A shipment's storage and transportation conditions can be monitored and tracked through the supply chain. Additionally, an OCW is used to communicate the latest shipment status with the external world. With Cumulus, parachains are able to send and receive cross-chain messages and enable validators to validate their state transitions.

[1]A cumulus is a dense, white, fluffy, flat-based cloud with a multiple rounded top and a well-defined outline. together they form a system that is intricate, beautiful and functional. The name was chosen because of the dot like shape (Polka-dot)

Chapter 6

Relay Tier

The relay tier is where the cross-chain messages are relayed from a source parachain to a target parachain. ChainFresh leverages the relay chain implementation of Polkadot, introduced in chapter 2.3.2 on page 13. First, section 6.1 briefly outlines the parachain registration. Next, section 6.2 continues the path from parablock creation to inclusion in the relay chain. The final section 6.3, describes cross-chain communication with one illustrative example for the cross-chain transfer of document existence proofs.

6.1 Parachain Registration

Before parachains can benefit from shared security and cross-chain communication, they need to be registered on the relay chain. The following rule is defined in the Collator Protocol [40], which implements the network protocol for the Collator-to-Validator networking: To accept n parachain connections, $n + 1$ validator nodes need to run on the relay chain. For the ChainFresh prototype, two relay chain nodes are started to connect one parachain node. Further, the relay chain needs to obtain the hex-encoded parachain's genesis state (exported from a collator node) and the Wasm runtime validation function to validate parablocks.

6.2 Parachain Block Inclusion

Chapter 3 describes the block production process in a parachain. This section continues the path from creation to the inclusion of a parablock into the relay chain. Appendix A depicts the six-step inclusion process described in the following.

1) The collator passes the collation to one of the relay chain validators. The validator must belong to the group of validators from the previous section (6.1) that were initally provided by the same parachain as the collator. A collation contains a candidate and a Proof of validation (PoV). The validator then passes the collation to the rest of the validators, provided by the same parchain, through the collator protocol.

2) The validators in the group verify and agree on candidates through candidate backing. Verification and consensus are a one-to-one correspondence. Only after the verification is successful, vote +1 (so after most of the validators vote) is it called consensus. Candidates that have collected enough signed and valid votes from other validators are considered "backable."

3) The block producer of the relay chain records both the backable candidate and the backing of the candidate in the block of the relay chain. A backable candidate is included in the block of the relay chain and is considered to be backed in the branch of the relay chain. (Each parachain will have at most one backable candidate, and the corresponding backing will be included in the block of the relay chain. At this time, the relay chain has not been confirmed.)

4) Once the candiatate is backed in the relay chain, the candidate is "pending available." It will not be considered as part of the parachain until it is proven to be available.

5) In the subsequent relay chain block, validators will make the candidate available through the availability distribution subsystem, and the information available about the candidate will be recorded in the subsequent relay chain block (more than two-thirds of all validators are held PoV of candidate).

6) Once the state machine of the relay chain (the validator's block producer) has enough information to believe that the candidate's PoV is available, then the candidate will be considered part of the parachain and can become a complete parachain block (step 7; only by including the information in the relay chain can there be complete data in step 8. The block producer of the relay chain packs the available candidates on the chain and then updates the head of the parachain and processes the messages contained in the candidate.)

6.3 Cross-Chain Asset Transfers

Parachains, and interoperability between them, is highly facilitated by ordered and timely delivery of messages. HRMP is the underlying protocol that ChainFresh parachains use to send messages to each other (i.e., facilitate cross-chain asset transfers). This section illustrates the process of transferring a document's existence proof from the source parachain of Organization A (OrgA) to the target parachain of organization B (OrgB).

6.3.1 Opening a Messaging Channel

OrgA can initiate a messaging channel to OrgB, depicted in figure 6.1. Therefore, OrgA must send an upward `Router::init_open_channel` message to the relay chain that includes the target parachain ID and the conditions for closing that channel (e.g., timeout).

This upward message will add an entry to OrgA's CST on the relay chain. In this table, every entry has information such as the recipient parachain ID, configurations such as the size or number of messages allowed on this channel, and the status of the channel, which can be on of "open," "pending open," "pending closing," or "closing." Here, the channel will be set to "pending open." Then, the relay chain sends a downward message to OrgB that includes the metadata (parachain ID of initiator, the sizes and number of messages) of the channel.

If OrgB is willing to have a channel open with the initiator parachain OrgB, it must send an upward `Router::accept_open_channel` message to the relay chain. This upward message includes the metadata it was sent earlier. Once the relay chain receives the message, it must match the metadata sent by OrgB's message and OrgA's CST entry that was created earlier for this purpose. If it does, the status in OrgA's CST entry corresponding to this channel is set to "open." If OrgB does not respond to the relay chain message, OrgA can cancel the channel initiation at any moment. However, some time must pass between initiation and cancellation to avoid race conditions.

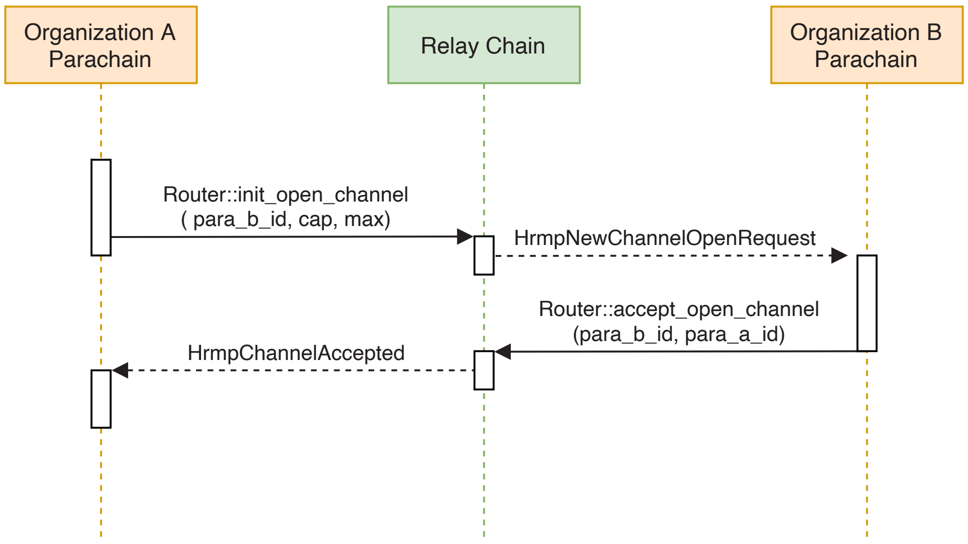

Organization A Parachain
Relay Chain
Organization B Parachain
Router::init_open_channel (para_b_id, cap, max)
HrmpNewChannelOpenRequest
Router::accept_open_channel (para_b_id, para_a_id)
HrmpChannelAccepted

Figure 6.1: HRMP opening sequence

6.3.2 Executing a Cross-Chain Transfer

The cross-chain transfer sequence starts when a messaging channel is established. OrgA can initiate a cross-chain transfer by putting a message with the file digest and owner into the messaging channel and calling `Router::queue_outbound_hrmp`. This will transfers OrgA's message outbound queue data to OrgB's message inbound queue. OrgB calls `Router::prune_hrmp` on its message entry queue and returns message data with a watermark (a relay chain block number indicating that incoming messages before the current block have been processed).

6.3.3 Closing a Messaging Channel

The closing sequence (shown in figure 6.2) begins when OrgA sends an upward message request to the relay chain for closure via `InitCloseChannel`. After this point, no new HRMP messages of OrgA's parachain are accepted for that channel. Then `Router::close_channel` is called, and the relay chain status of the channel becomes "pending closing." Afterward, the relay chain sends a downward message `HrmpChannelClosing` to OrgB to signal closing the messaging channel. OrgB can then end the channel with some `AcceptCloseChannel` message. Finally, the message queues are cleared, the channel is closed, and the metadata of the messaging channel is deleted.

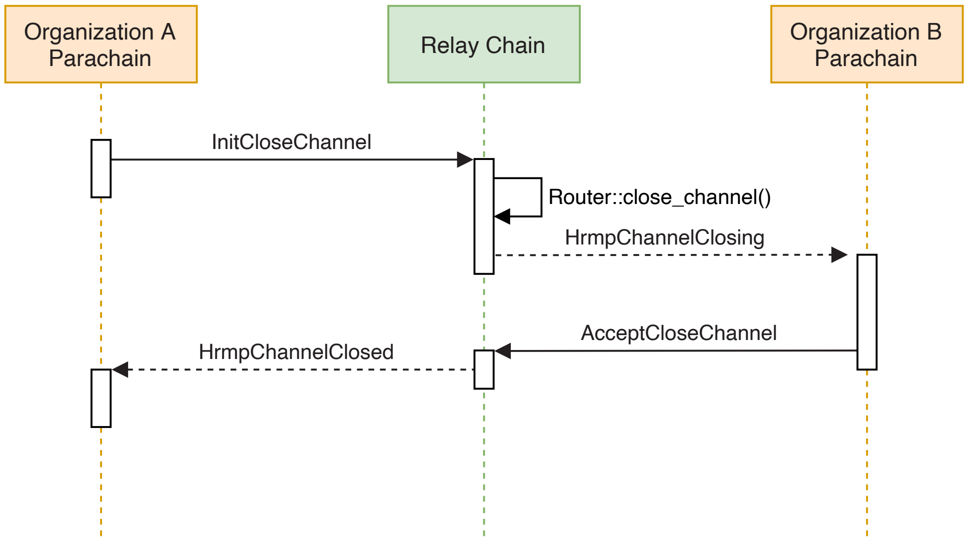


Figure 6.2: HRMP closing sequence

6.4 Summary

This chapter explained the relay tier in greater detail. Before a parachain can benefit from shared security and cross-chain communication, it must be registered at the relay chain. For shared security, the parachains make use of the relay chain block finalization consensus. Each candidate is first checked by the parachain validators, then by a randomly selected BABE relay chain validator, and finalized by the secondary checkers. For cross-chain messaging, a messaging channel must be established. When this happens, the messages (data) can be passed up from a source parachain to the relay chain. Subsequently, the relay chain passes the messages down to the target parachain. When all messages are processed and the messaging channel is no longer required, it is closed by the source parachain.

Part III

Conclusions

Chapter 7

Evaluation

This chapter provides a technical evaluation of the implemented ChainFresh system. The first section (7.1) evaluates cross-chain messaging. The second section (7.2) outlines the hard disk footprint of the system artifacts and accounts for the library dependencies of the web frontend. The final section (7.3) discusses current system limitations of the ChainFresh implementation.

7.1 Cross-Chain Messaging

To date, it seems that little or no effort has been put forward to evaluate a multichain and cross-chain communication, based on a literature review of the state of the art in blockchain evaluation [32, 10]. In the context of the ChainFresh system, this circumstance led to the formulation of RQ2: How can cross-chain transfers of a heterogeneous multichain be evaluated? This question is answered in the following sections: the basic idea of the evaluation approach is introduced in section 7.1.1, and the adopted equations from the field of fluid dynamics are outlined in section 7.1.2 and discretized to fit on-chain data in section 7.1.3. Finally, the numerical results are visualized in section 7.1.5.

7.1.1 Evaluation Approach

The evaluation of cross-chain messages suggests that the focus should be rather on finding macroscopic variables to capture the full system rather than microscopic variables concerning a single parachain. By adapting the perspective of fluid dynamics, observing a network of blockchains from a monitor, one can see the movement of cross-chain transfers as fluid.

Due to the heterogeneity of the data that can be moved between parachains, there is a need for a uniform measure unit. Typically, cross-chain messaging involves the transfer of tokens or arbitrary data. In major blockchain networks (e.g., Ethereum), Planck length[1] is chosen as a uniform unit of measure. A message is assigned one unit of transfer unit, if it is not a token transfer. Otherwise, the amount of transfer units equals the amount of tokens transferred in the message. The transfer volume (short: volume) of a parachain p is the total amount of units transferred in a given time interval. As stated in the previous chapter (6), a parachain can be uniquely identified in the network by its parachain ID.

There are two ways to describe the resulting fluid flow in physics: the Lagrangian specification and the Eulerian specification. In the Lagrangian specification, each fluid particle is tracked while it is moving in time and space, while the Eulerian specification focuses on a specific subspace and observes aggregate particles of fluid as they flow through subspace as time moves forward. For this approach, the Eulerian specification is chosen because parachains of similar transfer volume are aggregated into a subspace. This allows for capturing transfer volume irregularities in subspaces and over time.

7.1.2 Adapted Fluid Dynamics Equations

The equations introduced in this section are defined for the continuous domain.

Density

The first variable is the Density ρ. For example, in traffic flow models, $\rho(x,t)$ is the density of cars on a highway. That is the number of cars per kilometer at milepost x at time t [37]. Here, $\rho(x,t)$ is the number of parachains P per transfer volume at time t. Let $x \in \mathbb{R}^+$ represent the transfer volume of parachains, and let $\rho(x,t)$ be the density of parachains at transfer volume x and time t. Let $N(x,t)$ denote the total number of parachains in volume section $[x_0, x_1]$ (with $x_0 \leq x \leq x_1$) at time t. The following definition for density is obtained:

$$N(x,t) = \int_{x_0}^{x_1} \rho(x,t)dx \tag{7.1}$$

Velocity

The Velocity $u_p(t)$ of parachain p at time t is a measure of how quickly units are circulating in and out of a parachain. It is defined as:

$$u_p(t) = \lim_{\Delta t \to 0} \frac{v_p(t) - v_p(t - \Delta t)}{\Delta t} \tag{7.2}$$

where $v_p(t)$ is the transfer volume of parachain p at time t. The average velocity of parachain in volume section $[x_0, x_1]$ at time t is denoted by $v(x,t)$, which is constant for

[1]Planck length is a unit of length and the smallest possible distance in the physical universe. It was originally proposed by Max Planck, a German theoretical physicist (* April 23, 1858; † October 4, 1947).

each $x \in [x_0, x_1]$ and is defined as the average change in volume across all parachains in that volume section at time t. The average velocity function is given by

$$\bar{u}(x,t) = \frac{1}{N(x,t)} \sum_{k=1}^{N(x,t)} u_k(t) \tag{7.3}$$

Rate of Flow

Assuming that parachains are neither added nor removed from the network, the number of parachains can change only as a result of flow across the endpoints x_0, x_1. The rate of flow (called the Flux) of parachains is given by

$$Q(x,t) = \rho(x,t)v(x,t) \tag{7.4}$$

7.1.3 Discretization

Adapting the perspective of fluid dynamics is only the first step toward a quantitative evaluation approach for cross-chain messaging. The equations in section 7.1.2 are defined for the continuous domain and require modification to apply to discrete blockchain data. We discretize these equations as follows. Consider a set of parachains K, each of which has a transfer volume measured at regular times, each Δt apart. For a set of T intervals, the time intervals are indexed by $t \in 0, 1, ..., T$. Figure 7.1 depicts how the domain of volume x and time t is discretized into regions $[x_0, x_1] \times [t_0, t_1]$, where $[x_0, x_1]$ is a volume range and $[t_0, t_1]$ is a time interval.

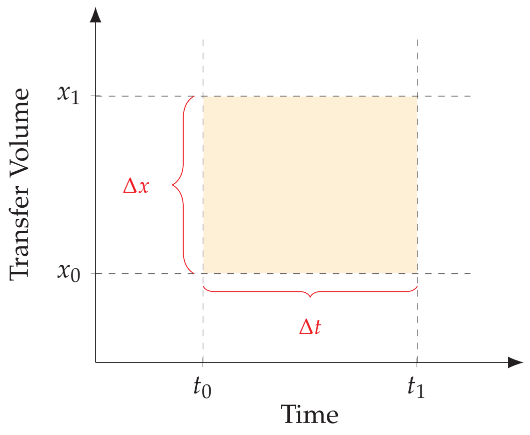


Figure 7.1: Volume–time discretization

The transfer volume domain is partitioned into sections. In this analysis, a volume discretization of $\Delta x = 10$ is applied. The value for Δx was chosen empirically. The transfer volume section denoted by $S_x, x \in \mathbb{Z}^+$ contains parachains in the interval $[x_i, x_{i+1}]$. The discrete approximation of N is:

$$N(x,t) = |\{k \in K \mid v_k(t) \in S_x\}| \tag{7.5}$$

The density can be approximated by ratio between the number of parachains in a volume section and the section area.

$$\rho(x,t) = \frac{N(x,t)}{\Delta x \Delta t}. \tag{7.6}$$

The velocity of a parachain can be approximated by the first derivative of its transfer volume with respect to time:

$$u_k(t) = \frac{v_k(t) - v_k(t - \Delta t)}{\Delta t} \quad k \in K \tag{7.7}$$

Thus, the average velocity of parachains in a transfer volume section $\bar{u}(x,t)$ can be approximated by the sum of velocities over the total sum of parachains in this section.

$$\bar{u}(x,t) = \frac{1}{N(x,t)} \sum_{p_k(t) \in S_x} u_k(t) \quad x \in \mathbb{Z}^+ \tag{7.8}$$

7.1.4 Evaluation Workflow

To apply the evaluation approach, block resolution on-chain data is collected from a full node. A full node is a relay chain validator node. The block resolution refers to the block production time of six seconds, described in chapter 5. The evaluation workflow, depicted in figure 7.2, is divided into the following six steps:

1) *Initalization*: The workflow requires a connection to the running ChainFresh system, which it can monitor for new block headers. Thus, the first step of the workflow establishes a websocket connection to a full node.
2) *Monitoring*: In this step, the running system is observed on new block headers. When a new block header becomes available, the `subscription_handler` hands over the parent hash of the new block header for the following pre-processing steps.

The data pre-processing workflow, in steps 3 and 4, is adopted from Framewala et al. [12] and modified for real-time analysis (monitoring).

3) *Extrinsics Deserialization*: With the parent hash, the extrinsics contained in a block can be fetched and deserialized.
4) *Unit Conversion*: Extrinsics can contain information about transfer amounts (e.g., balance transfers) or not in the case of document transfers. Therefore, each extrinsic gets transfer units assigned in Planck length (10^{-8}).

5) *On-Chain Data Serialization*: After fetching the latest blockchain data, it must be parsed to convert it to a processable format in a structured form. The parser is used to make transactions from all the blocks available in a readable format. The processed on-chain data is stored in a structured format having fields such as timestamp, source parachain, target parachain, and transfer value. 6) *Visualization*: Finally, the equations from the previous section (7.1.3) and section 7.1.2 respectively are applied to the collected on-chain data. The resulting graph plots are generated with Matplotlib. The results are discussed in the next section 7.1.5.

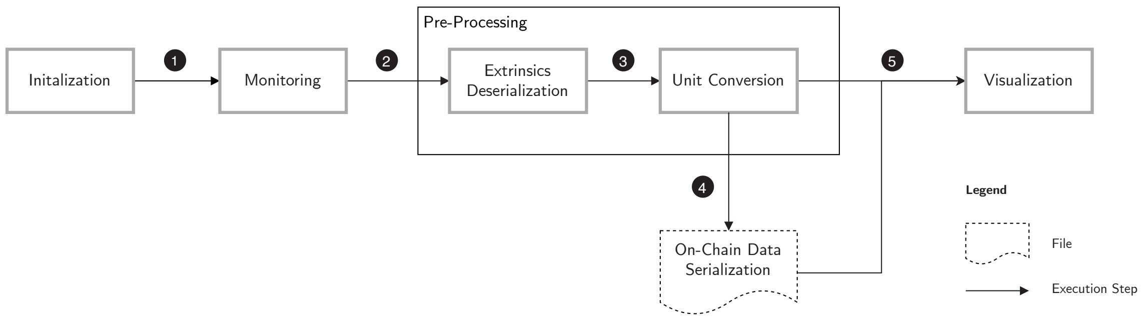


Figure 7.2: Test workflow

7.1.5 Numerical Results

The cross-chain messages in this setup were partly manually generated via the GUI and automatically using an XCM generator. The XCM generator is not part of the evaluation workflow as it is only used to simulate realistic transfer load.

Three plots of the variable velocity for different transfer volume ranges are shown in figure 7.3. This figure reflects considerable variability and sensitivity around the minutes 180 to 200, as the velocity (blue) deviates from the mean velocity (red). Notably, spikes in velocity were only measured in the low transfer volume range (< 10 units) and the medium transfer volume range (10 to 100 units). Once unusual variability in transfer volume is detected, one can explore the involved parachains and possible causes.

It must be expected that the same results cannot be achieved as easily with the same workflow when XCMP is available. Using XCMP, the payload will not be stored on the relay chain anymore. Instead, one must connect to at least one collator node of each parachain to access the payload of a cross-chain message.

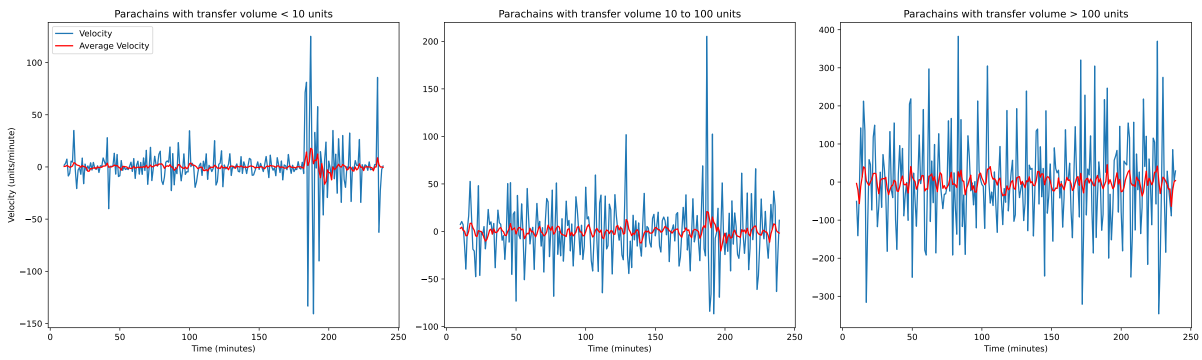


Figure 7.3: Transfer volume velocities

7.2 Hard Disk Footprint

The four artifacts of ChainFresh are the `polkadot` executable for the relay tier, the `parachain` executable and the `ocw` script for the application tier, and a `build` folder for the presentation tier. All four artifacts, each with its build tool and artifact size, are listed in table 7.1.

System name	Build tool	Artifact name	Artifact size
Frontend	Yarn	build	19.1 MB
Off-Chain Service	Python Interpreter	ocw.py	414 B
Parachain	Rust Compiler	parachain.exec	43.6 MB
Relaychain	Rust Compiler	polkadot.exec	162.5 MB

Table 7.1: ChainFresh artifacts

To generate the web frontend build folder with static files for production, Yarn [44] bundles several external libraries along the frontend files. The primary external library is `react` for building the browser-based ChainFresh user interface. The `react-dom`, `react-router-dom`, and `react-scripts` dependencies are part of React. The `carbon-components` library is the official implementation of the Carbon Design System.[2] Carbon components are a collection of reusable HTML and SCSS partials to ease the development of the browser-based user interfaces. The `sass` library is used for the SCSS partials of the carbon components. The `xxhashjs` library is for cryptographic hashing purposes. The web frontend uses it for client-side hashing of the files to be used by the document registry in the application tier. The `substrate-lib` submodule makes use of the `prop-types` library to ensure the right properties are passed to the transaction button and the `query-string` library to parse the connected websocket string. The `@polkadot` package includes `extension-dapp` to retrieve all the Web3-injected providers added to the web page; `types` provides helper methods to convert a type returned by the API from or to Codec; `ui-keyring` to manage the state of a connected blockchain account in storage; `ui-settings` to manage the API endpoints for the selected blockchain; `util` and `util-crypto` to provide utility functions with additional safety checks for consistent coding; and `api` to provide a wrapper around all the methods exposed by a Substrate client in the application tier. All frontend dependencies, each with its version and file size, are listed in table 7.2.

[2]Carbon Design System implements the IBM Design Language. The designs are systemic and logical as they all follow the same universal principles. Initial public development started on June 10, 2015.

Dependency	Version	File size
@carbon/colors	10.34.0	160 kB
carbon-components	10.25.0	8.5 MB
carbon-components-react	7.25.0	6.7 MB
carbon-icons	7.0.7	902 kB
@carbon/icons-react	10.42.0	55 MB
@polkadot/api	1.34.1	570 kB
@polkadot/extension-dapp	0.34.1	26 kB
@polkadot/types	1.34.1	1.1 MB
@polkadot/ui-keyring	0.58.1	89 kB
@polkadot/ui-settings	0.58.1	36 kB
@polkadot/util	3.4.1	416 kB
@polkadot/util-crypto	3.4.1	529 kB
prop-types	15.7.2	104 kB
query-string	6.13.2	32 kB
react	16.13.1	204 kB
react-dom	16.13.1	3 MB
react-router-dom	5.2.0	723 kB
react-scripts	3.4.3	11.9 MB
sass	1.29.0	4.3 MB
xxhashjs	0.2.2	182 kB

Table 7.2: Frontend dependencies table

7.3 System Limitations

The implementation of ChainFresh is a proof of concept and has several limitations. The project uses Cargo [8] to download and compile the Rust package's dependencies for the relay chain and the parachains. The build time for `polkadot` took 1705.39 seconds in one build. The build time for the `parachains` took 940.11 seconds on average in eight builds. For the web frontend, Yarn generated a production build folder in 68.07 seconds on average across seven builds. The total average system build time is 2713 seconds. The measurements were obtained with a 6-Core Intel Core i7, 32 GB RAM local machine during system implementation. Due to the limited number of builds on one local machine, the results must to be interpreted with caution. Different hardware will likely yield different system build times.

One possible way to accelerate the build time, if no superior hardware configuration is available locally, is to use Cargo Remote [20]. In the context of this thesis, a virtual machine was set up on Amazon Web Services with 16 virtual CPUs and 64 GB memory. The build time for `polkadot` took 510.09 seconds in one build, similar to the results of Kopp [20]. The build time for a `parachain` took 182.13 seconds.

Currently, ChainFresh is leveraging the Polkadot relay chain implementation with HRMP for cross-chain messaging. Consequently, the relay chain contains configurations and packages not used by ChainFresh, resulting in slower build times. Furthermore, with HRMP, the full payload of a message is stored in the relay chain. While this allows for the evaluation approach in section 7.1, it constrains the benefit of controlled transparency in section 3.1.

Room for improvement involves the organization selector (section 4.3). Conceptually, the organization selector accounts for the fact that an organization could be divided into multiple divisions. Thus, an account could be part of multiple divisions or roles in the same organizations. The registrar pallet maintains a list of organizations and maps each organization to a member in a one-to-one relationship. Organizations are identified by the account ID that registered it. This limits an account to register at most one organization. Simultaneously, one organizations cannot be mapped to multiple accounts.

7.4 Summary

This chapter provided a technical evaluation of the ChainFresh system. Initially, the concepts of density, velocity, and flux were adapted from the field of fluid dynamics and introduced to capture the variability of the transfer volume in cross-chain messaging. The characteristics of the evaluation approach define it as performance monitoring. Next, the executable artifacts have been described with their hard disk footprint. The included external library dependencies for the build folder of the web frontend have also been accounted for. Finally, current system limitations of the ChainFresh implementation were discussed.

Chapter 8

Future Work

ChainFresh was designed with extensibility and modularity in mind. Several ideas for extensions and modifications of the current ChainFresh system are presented here.

Section 2.3.1 introduced the network of heterogeneous state machines approach, along with the unified state machine with heterogeneous shards approach implemented in this thesis. Both approaches belong to the BIF category BoB. Future work may conduct a comparative analysis between the two approaches. Methods to achieve this include the substitution of the XCMP protocol with the IBC protocol. A runtime pallet called `pallet-ibc` (version 2.0.0-pre.2) [36], which implements the standard IBC protocol for substrate-based blockchains, is currently under development. This pallet would replace the `cumulus` pallet in the application tier, effectively converting parachains into zones. In the relay tier, the relay chain would be replaced by one or more hubs. Still, the terminology stays the same. Parachains and zones both implement application-specific logic. The relay chain and hubs are both relayers, outlined in the state-of-art section (2.3.1). Therefore, the three-tier system architecture as a whole remains unaffected by such change.

A way to extend the current ChainFresh system is to introduce further functionalities for products, shipments, and documents. For instance, if a user wants to update an existing document, a new dispatchable function, `update_document`, must be introduced in the `DocumentRegistry` pallet. The document can then be updated by calling the new dispatchable function with parameters `title` of type `<Vec<DocumentTitle>>` to change the existing document tile and `organization` of type `DocumentOrganization` to share the document with more organizations than the one initially selected. Analogously, all other pallets can be extended with additional functionality.

Chapter 9

Conclusion

Developing long-term and increasingly collaborative relationships among supply chain participants requires advanced technological solutions to retain a competitive edge. Blockchain is presented as a promising technology that might increase supply chain visibility and improve efficiency. This thesis offers an answer for both research questions stated at the beginning.

RQ1: How can a blockchain-enabled supply chain network be designed with interoperability and controlled transparency? This question is answered with a reference implementation named ChainFresh, which implements the multi-chain approach for blockchain-enabled supply chains. The ChainFresh system architecture is composed of three tiers: presentation, application, and relay. The Presentation tier provides the user with convenient client-side access to the system. The application tier encompasses each organization's parachain, which is responsible for realizing the core business logic of ChainFresh. In turn, this allows for controlled transparency on a blockchain level as each organization has control over the data it shares. Finally, interoperability is realized by the relay tier through leveraging the Polkadot relay chain implementation and XCMP for cross-chain communication.

RQ2: How can cross-chain message passing be quantitatively evaluated in a heterogeneous multi-chain? The concepts of density, velocity, and flux were adapted from the field of fluid dynamics (Eulerian specification) and introduced to capture the variability of the transfer volume in cross-chain messaging.

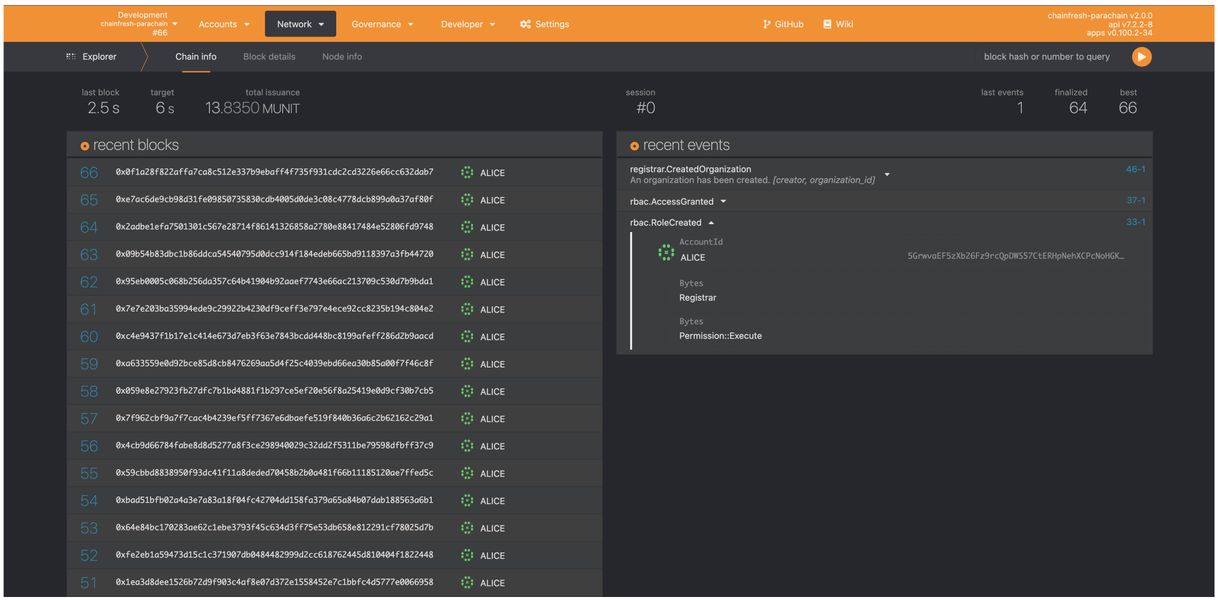
The work primarily followed the methodology of the design science approach by Hevner [19] to produce the ChainFresh system artifacts. proves the feasibility of the ChainFresh concept. The ChainFresh code is available in the Cyberlytics repository: `https://github.com/cyberlytics/ChainFresh`.

Finally, the design approach of ChainFresh can benefit more than just the supply chain industry. In essence, it can apply to any network that requires the transfer of sensitive data. However, blockchain technology is still at its infancy stage, and there is still a long way before it can be widely put into use. For this to occur, supply chain organizations must not only adopt the new technology but also use it as an information-sharing medium.

Appendices

Appendix A

ChainFresh System



last block 2.5 s | target 6 s | total issuance 13.8350 MUNIT | session #0 | last events 1 | finalized 64 | best 66

recent blocks

```
66  0x0f1a28f822affa7ca8c512e337b9ebaff4f735f931cdc2cd3226e66cc632dab7  ALICE
65  0xe7ac6de9cb98d31fe09850735830cdb4005d0de3c08c4778dcb899a0a37af80f  ALICE
64  0x2adbe1efa7501301c567e28714f86141326858a2780e88417484e52806fd9748  ALICE
63  0x09b54b83dbc1b86ddca54540795d0dcc914f184edeb665bd9118397a3fb44720  ALICE
62  0x95eb0005c068b256da357c64b41904b92aaef7743e66ac213709c530d7b9bda1  ALICE
61  0x7e7e203ba35994ede9c29922b4230df9ceff3e797e4ece92cc8235b194c804e2  ALICE
60  0xc4e9437f1b17e1c414e673d7eb3f63e7843bcdd448bc8199afeff286d2b9aacd  ALICE
59  0xa633559e0d92bce85d8cb8476269aa5d4f25c4039ebd66ea30b85a00f7f46c8f  ALICE
58  0x059e8e27923fb27dfc7b1bd4881f1b297ce5ef20e56f8a25419e0d9cf30b7cb5  ALICE
57  0x7f962cbf9a7f7cac4b4239ef5ff7367e6dbaefe519f840b36a6c2b62162c29a1  ALICE
56  0x4cb9d66784fabe8d8d5277a8f3ce298940029c32dd2f5311be79598dfbff37c9  ALICE
55  0x59cbbd8838950f93dc41f11a8deded70458b2b0a481f66b11185120ae7ffed5c  ALICE
54  0xbad51bfb02a4a3e7a83a18f04fc42704dd158fa379a65a84b07dab188563a6b1  ALICE
53  0x64e84bc170283ae62c1ebe3793f45c634d3ff75e53db658e812291cf78025d7b  ALICE
52  0xfe2eb1a59473d15c1c371907db0484482999d2cc618762445d810404f1822448  ALICE
51  0x1ea3d8dee1526b72d9f903c4af8e07d372e1558452e7c1bbfc4d5777e0066958  ALICE
```

recent events

```
registrar.CreatedOrganization                                      46-1
An organization has been created. [creator, organization_id]
rbac.AccessGranted                                                 37-1
rbac.RoleCreated                                                   33-1
    AccountId
    ALICE        5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGK…
    Bytes
    Registrar
    Bytes
    Permission::Execute
```

Figure A.1: Polkadot{.js} UI Blockchain explorer. On the left hand side, recent blocks with their author are listed. On the right hand side, recent events inside recent blocks are listed.

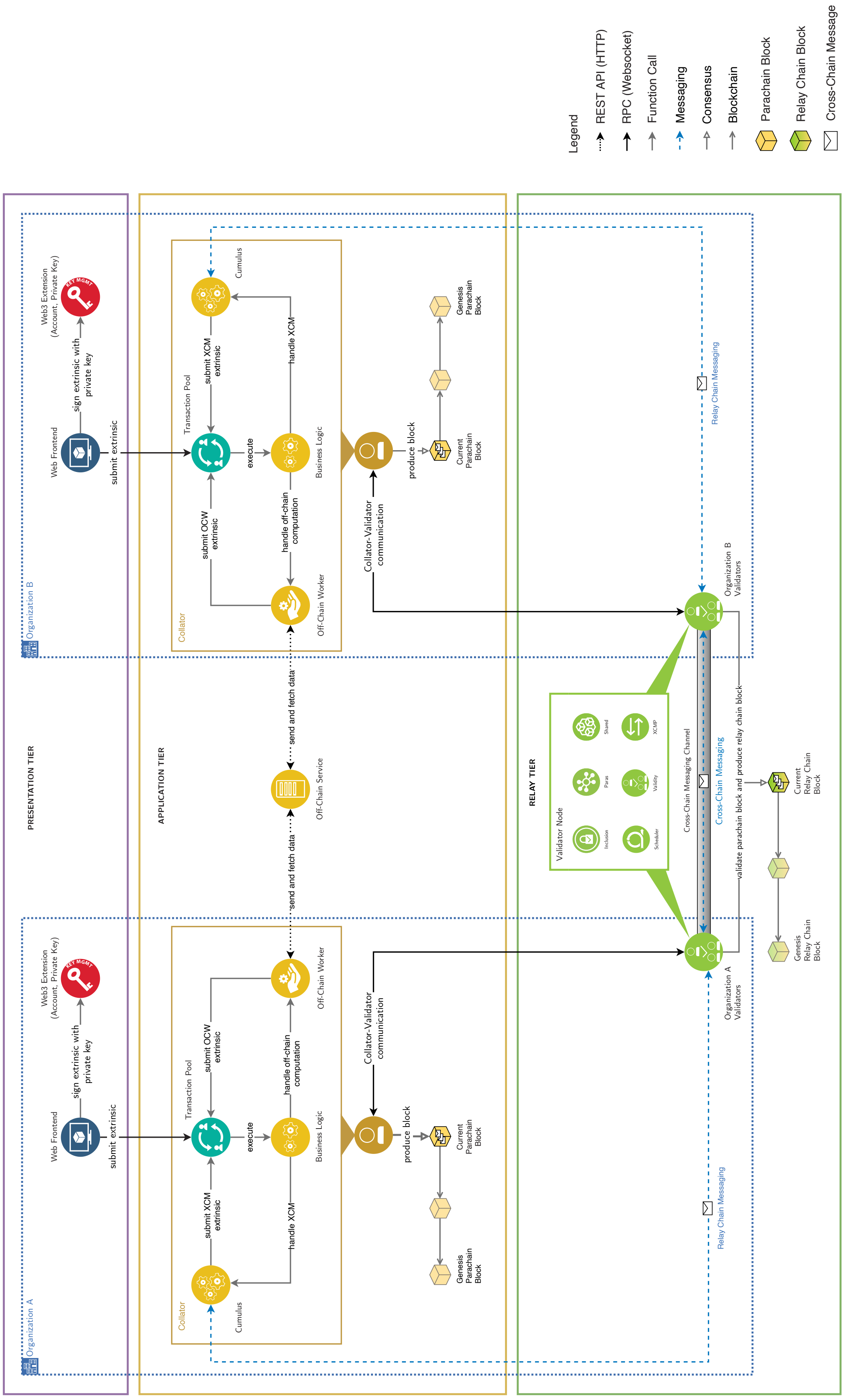


Figure A.2: Holistic ChainFresh system design overview for two interoperating organizations.

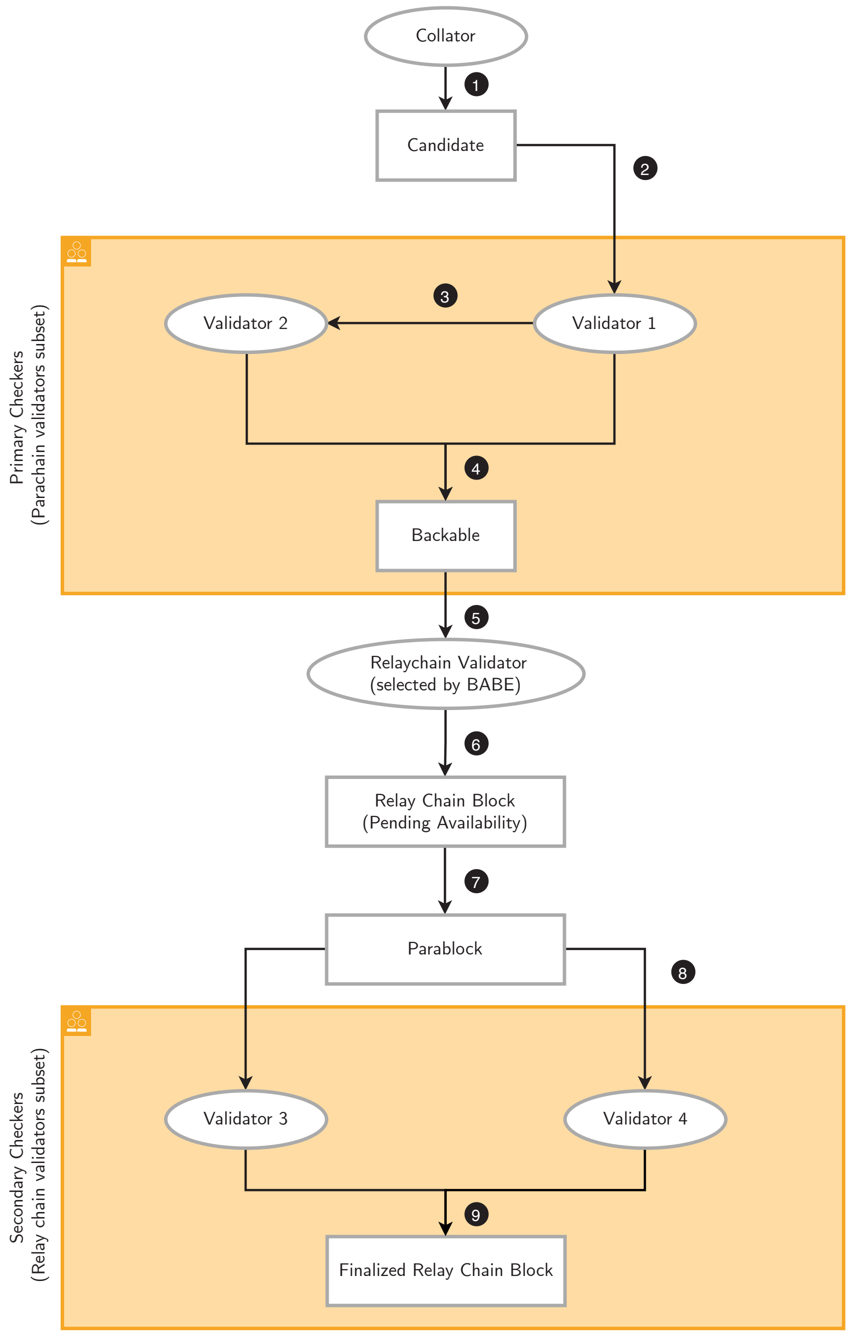


Figure A.3: Block inclusion process

Bibliography

[1] Handan Kilinc Alper. *Blind Assignment for Blockchain Extension Protocol,* [Online]. 2021. URL: https://research.web3.foundation/en/latest/polkadot/block-production/Babe.html.

[2] Andreas M. Antonopoulos and Gavin Wood. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[3] APQC. *Blockchain adoption in supply chain: Current state for 2020*. [Online]. 2020. URL: https://www.apqc.org/system/files/resource-file/2020-05/K010594_SCM%20BlockChain%20IG_QuickPoll13.pdf.

[4] Rafael Belchior et al. "A Survey on Blockchain Interoperability: Past, Present, and Future Trends". In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–41.

[5] Michael A. Bourlakis and Paul W. H. Weightman. *Food supply chain management*. John Wiley & Sons, 2008.

[6] Mic Bowman et al. "Private Data Objects: an Overview". In: *ArXiv* abs/1807.05686 (2018).

[7] Jeff Burdges et al. "Overview of polkadot and its design considerations". In: *ArXiv* abs/2005.13456 (2020).

[8] *Cargo*. [Online]. 2021. URL: https://doc.rust-lang.org/cargo/.

[9] Niklas Egels-Zandén, Kajsa Hulthén, and Gabriella Wulff. "Trade-offs in supply chain transparency: the case of Nudie Jeans Co". In: *Journal of Cleaner Production* 107 (2015), pp. 95–104.

[10] Caixiang Fan et al. "Performance evaluation of blockchain systems: A systematic survey". In: *IEEE Access* 8 (2020), pp. 126927–126950.

[11] David Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli. *Role-based access control*. Artech house, 2003.

[12] Aman Framewala et al. "Blockchain Analysis Tool For Monitoring Coin Flow". In: *2020 Seventh International Conference on Software Defined Systems (SDS)*. IEEE. 2020, pp. 196–201.

[13] Erich Gamma et al. *Design patterns: Elements of reusable software architecture*. 1995.

[14] Christopher Goes. "The Interblockchain Communication Protocol: An Overview". In: *ArXiv* abs/2006.15918 (2020).

[15] GS1. *GS1 GSIN*. [Online]. 2015. URL: https://www.gs1.org/docs/idkeys/GS1_GSIN_Executive_Summary.pdf.

[16] GS1. *GS1 GTIN*. [Online]. 2015. URL: https://www.gs1.org/docs/idkeys/GS1_GTIN_Executive_Summary.pdf.

[17] Haya Hasan et al. "Smart contract-based approach for efficient shipment management". In: *Computers & Industrial Engineering* 136 (2019), pp. 149–159.

[18] Christine V. Helliar et al. "Permissionless and permissioned blockchain diffusion". In: *International Journal of Information Management* 54 (2020), p. 102136. ISSN: 0268-4012. DOI: https://doi.org/10.1016/j.ijinfomgt.2020.102136. URL: https://www.sciencedirect.com/science/article/pii/S0268401219314586.

[19] Alan Hevner. "A Three Cycle View of Design Science Research". In: *Scandinavian Journal of Information Systems* 19.2 (2007), p. 4.

[20] Wilfried Kopp. *Cargo Remote: Speed up your Rust builds*. [Online]. 2019. URL: https://www.chevdor.com/post/2019/12/23/cargo-remote/#_enjoy_the_powaaa.

[21] Rick Kuhn, Dylan Yaga, and Jeffrey Voas. "Rethinking distributed ledger technology". In: *Computer* 52.2 (2019), pp. 68–72.

[22] Jae Kwon. "Tendermint: Consensus without mining". In: *Draft v. 0.6, fall* 1.11 (2014).

[23] Jae Kwon and Ethan Buchman. *Cosmos Whitepaper*. [Online]. 2019. URL: https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md.

[24] *libP2P - A modular network stack.* [Online]. 2021. URL: https://libp2p.io/.

[25] Francesco Longo et al. "Blockchain-enabled supply chain: An experimental study". In: *Computers & Industrial Engineering* 136 (2019), pp. 57–69.

[26] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[27] Shenle Pan, Ray Y. Zhong, and Ting Qu. “Smart product-service systems in interoperable logistics: Design and implementation prospects”. In: *Advanced Engineering Informatics* 42 (2019), p. 100996.

[28] Parity. *Off-Chain Features*. [Online]. 2021. URL: `https://docs.substrate.io/v3/concepts/off-chain-features/`.

[29] Aditi Rao, Blythe Hurley, and Rupesh Bhat. “From siloed to distributed: Blockchain enables the digital supply network”. In: (2019).

[30] *RocksDB - A persistent key-value store for fast storage environment*. [Online]. 2021. URL: `http://rocksdb.org/`.

[31] Kai Fabian Schulz and Daniel Freund. “A Multichain Architecture for Distributed Supply Chain Design in Industry 4.0”. In: *International Conference on Business Information Systems*. Springer. 2018, pp. 277–288.

[32] Sergey Smetanin et al. “Blockchain Evaluation Approaches: State-of-the-Art and Future Perspective”. In: *Sensors* 20.12 (2020), p. 3358.

[33] Manu Sporny et al. *Decentralized Identifiers (DIDs) v1.0: Core architecture, data model, and representations - W3C Working Draft 03 August 2021*. [Online]. 2021. URL: `https://w3c.github.io/did-core/`.

[34] Alistair Stewart and Eleftherios Kokoris-Kogia. “GRANDPA: a Byzantine Finality Gadget”. In: *ArXiv* abs/2007.01560 (2020).

[35] *Substrate - The Blockchain Framework for a Multichain Future*. [Online]. 2021. URL: `https://substrate.io/`.

[36] *Substrate IBC Pallet*. [Online]. 2021. URL: `https://crates.io/crates/pallet-ibc`.

[37] Martin Treiber and Arne Kesting. “Traffic flow dynamics”. In: *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013).

[38] USDA. *Grades and Standards of Strawberries*. [Online]. 2021. URL: `https://www.ams.usda.gov/grades-standards/strawberries-grades-and-standards`.

[39] Jack Van der Vorst, Adrie Beulens, and Teris van Beek. “Innovations in logistics and ICT in food supply chain networks”. In: *Innovation in Agri-Food systems* (Jan. 2005).

[40] W3F. *Collator Protocol*. [Online]. 2021. URL: `https://w3f.github.io/parachain-implementers-guide/node/collators/collator-protocol.html`.

[41] Shangping Wang et al. “Smart Contract-Based Product Traceability System in the Supply Chain Scenario”. In: *IEEE Access* 7 (2019), pp. 115122–115133.

[42] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[43] Gavin Wood. *Polkadot: Vision for a heterogeneous multi-chain framework*. [Online]. 2016. URL: `https://polkadot.network/PolkaDotPaper.pdf`.

[44] *Yarn - Safe, stable, reproducible projects*. [Online]. 2021. URL: `https://yarnpkg.com/`.

[45] Zibin Zheng et al. “An overview of blockchain technology: Architecture, consensus, and future trends”. In: *2017 IEEE International Congress on Big Data*. IEEE. 2017, pp. 557–564.

Glossary

Name	Description	Symbol (plural)	Def
BABE	*Blind Assignment of Blockchain Extension* is a block authoring mechanism to assign elected validators randomly for a certain block production slot.		2.3
BFT	*Byzantine Fault Tolerance* is the feature of a distributed network to reach consensus even when some of the nodes in the network fail to respond or respond with incorrect information.		2.3
BIF	*Blockchain Interoperability Framework*		2.0
BoB	*Blockchain of Blockchains*		2.0
CST	*Channel State Table*		6.3
Density	The number of parachains P at a given time t in a transfer volume section.	ρ	7.1
DID	A *Decentralized Identifier* is a globally unique and persistent identifier. No centralized registration authority is required because the identifier is registered on a distributed ledger and proved using cryptography.		5.2
DLT	*Distributed Ledger Technology*		1.1
Flux	The rate of flow of parachains.	Q	7.1
FRAME	*Framework for Runtime Aggregation of Modularized Entities* enables developers to create blockchain runtime environments from a modular set of components called pallets.		3.2
FSCN	*Food Supply Chain Network*		1.1
GRANDPA	*GHOST-based Recursive Ancestor Deriving Prefix Agreement*		2.3

Name	**Description**	**Symbol** (plural)	**Def**
GSIN	The *Global Shipment Identification Number* is a number assigned by a seller and shipper of goods to identify a shipment comprised of one or more logistic units that are intended to be delivered together.		4.4
GTIN	The *Global Trade Item Number* can be used by a company to uniquely identify all of its trade items.		4.3
GUI	*Graphical User Interface*		1.3
HRMP	*Horizontal Relay-routed Message Passing*		2.3
IBC	*Interblockchain Communication Protocol*		2.3
IDE	*Integrated Development Environment*		3.3
OCW	The *Off-Chain Worker* subsystem allows execution of long-running and possibly non-deterministic tasks (e.g., web requests) which could otherwise require longer than the block execution time.		3.2
P2P	*Peer-to-Peer*		1.1
PDO	*Private Data Object*		1.2
PoV	*Proof of validation*		6.2
RBAC	*Role-based Access Control*		3.2
RPC	*Remote Procedure Call*		4.0
UML	*Unified Modeling Language*		1.4
Velocity	The speed at which units are exchanged by a parachain in a cross-chain transfer.	u	7.1
Wasm	*WebAssembly* can be compiled from many languages, including the Rust programming language. Substrate-based chains use a WebAssembly binary to provide portable runtimes that can be included as part of the chain's state.		5.1
XCM	*Cross-Chain Messaging*		3.2
XCMP	*Cross-Chain Message Passing* is a protocol that parachains use to send messages to each other.		2.3

List of Figures

List of Tables