# Trustworthy Decentralized Last-Mile Delivery Framework Using Blockchain

Ala' Alqaisi[1], Sherif Saad[1] and Mohammad Mamun[2]
[1]*University of Windsor, Windsor, Canada*
[2]*National Research Council Canada, New Brunswick, Canada*

Keywords: Last-Mile Delivery, Crowd-Shipping, Blockchain, Reputation Models.

Abstract: The fierce competition in eCommerce is a painful headache for logistics companies. In 2021, Canada Post's parcel volume peaked at 361 million units with a minimum charge of $10 per each. Last-Mile Delivery (LMD) is the final leg of the supply chain that ends with the package at the customer's doorstep. LMD is the most costly process, accounting for more than 50% of the overall supply chain cost. Platforms such as Uber Eats and Amazon Flex help overcome this inefficiency and provide an outstanding delivery experience by enabling crowd-shipping using freelancer drivers willing to deliver packages in exchange for compensation. However, the current generation of LMD platforms that leverage crowd-shipping are centralized platforms and behave as intermediaries that charge commission fees. They lack transparency, and most of them, if not all, are platform monopolies in the making. This paper introduces the design of the next-generation LMD crowd-shipping platforms by leveraging Blockchain and smart contracts. A decentralized crowd-shipping platform for LMD that is scalable, reliable, secure and promotes fairness. The proposed platform connects the primary stakeholders of LMD without intermediaries. The stakeholders could use the platform to manage shipping and delivery, handle disputes, maintain fairness, and mitigate monopoly power. Our approach replaces the need for a centralized intermediator such as Uber by introducing a decentralized reputation model that executes over a cryptocurrency-less blockchain network. Our proof-of-concept implementation of the proposed framework demonstrates the potential of blockchain technology to decentralize the crowd economy. We used informal security analysis to illustrate how the proposed decentralized reputation model discourages misuse and encourages fairness between parties.

## 1 INTRODUCTION

Last-mile delivery (LMD) refers to the last segment of the supply chain process that transfers goods from final distribution centers to customers' doorstep. Due to consumers' dispersed destination locations, last-mile delivery is the most expensive and challenging stage of the delivery process (Dolan, 2023). In the last decade, the tremendous evolution of online shopping and the recent pandemic have been causing spectacular growth in the parcel shipping market across the globe. For instance, Canadian e-Commerce sales, which made up over $43 billion in 2018, are projected to increase by another 25% by 2023 to reach $55.4 billion (Le, 2022). As a result, Canada spent over $19.9 billion on last-mile delivery services in 2021 (Placek, 2022). Customers' preferences and expectations have increased along with the growth of e-Commerce; they now require same-day shipping, free home delivery, real-time package tracking, and free returns. These new conditions pressure shipping professionals to consider alternative LMD solutions.

Meanwhile, the world has witnessed a rapid evolution of the sharing economy, which refers to a peer-to-peer activity of obtaining, giving, or sharing access to goods and services through e-applications. Logistics providers exploited the same concept to develop a new business model called Crowd-shipping. It delegates parcel delivery tasks to a crowd of local couriers for monetary compensation using their vehicles or other transportation modes. The most common example of crowd-shipping LMD platforms is within the food and restaurant industry, such as Uber Eats, SkipTheDishes, DoorDash, and Instacart. Crowd-shipping maximizes logistics efficiency by downsizing the operational costs of package delivery, enhancing customers' flexibility to schedule deliveries with online shipment tracking, and reducing traffic and



Alqaisi, A., Saad, S. and Mamun, M.
Trustworthy Decentralized Last-Mile Delivery Framework Using Blockchain.
DOI: 10.5220/0012090300003552
In *Proceedings of the 20th International Conference on Smart Business Technologies (ICSBT 2023)*, pages 54-65
ISBN: 978-989-758-667-5; ISSN: 2184-772X

emissions (Gatta et al., 2018).

However, while these applications make life easier for the customer, they make it harder for others. To begin with, the organizations that run these applications act as intermediaries and deduct unjustifiable high commission rates for managing the delivery process between the retailer and the buyer. For instance, SkipTheDishes and UberEats commission rates range between 20% and 30% of each order's value (Evans, 2020). Secondly, giant companies' rivals and investments created a trend toward a monopoly and induced an uneven distribution of the welfare produced in the crowd-sourced delivery field, which forced small businesses to consider whether they could afford to continue playing in the delivery sector. Furthermore, most of these platforms are deployed on a centralized architecture, which could expose the system to data corruption, privacy breach, and single-point-of-failure risks, resulting in monetary and credit damage. For instance, DoorDash was a victim of hackers who stole the information of 4.9 million users (Breen, 2019), while similar incidents happened with UberEats in 2020 (N, 2020).

From this perspective, there is a call to transform the present platforms or construct an alternative business model that overcomes these drawbacks. The possibility to respond to this call is the Blockchain, which may potentially make a vital contribution to the supply chain and logistics field due to its critical features, such as immutability, integrity, and confidentiality. Blockchain can create trusted decentralized applications and reduce their reliance on third parties by utilizing smart contracts that define an automatic agreement between seller, courier, and consignee when transacting. Plus, distributed ledger guarantees transparency and avoids the risk of data tampering and infrastructure failure. Hence, this paper aims to design a blockchain-based crowd-shipping platform managed by retailers, customers, and couriers transparently and without intermediaries.

The main contributions of this paper can be described as follows:

- Propose a blockchain-based system to realize decentralized crowd-shipping services that eliminate the need for a central authority or third-party involvement.
- Propose a reputation model for couriers based on their prior behaviours to inject trust and discourage malicious behaviour in the system.
- Implement a proof of concept using Hyperledger Fabric, a real-world permissioned blockchain platform and conduct intensive experiments and performance evaluations in a test network.

The rest of this paper is organized as follows. Section 2 discusses related work in the literature. Section 3 briefly overviews the Hyperleger Fabric blockchain and the transaction flow, as it is the development environment of the subject in this study. Section 4 presents the proposed blockchain-based last-mile delivery framework. Then, section 5 describes a proof of concept implementation of the proposed framework and the development environment. Next, section 6 presents experimental results and discusses the security analysis of the implemented solution. Finally, section 7 concludes the paper with some future work.

## 2 RELATED WORK

In the literature, several works have proposed that leveraging blockchain technology in one way or another improves the delivery of physical assets such as parcels and packages. Here we concentrate on work that focuses on last-mile delivery.

A blockchain-based shipping platform called Lelantos was proposed in (AlTawy et al., 2017) to hide the customers' identity from shippers and prevent linkability between the customer and the merchant. Customers upload their real addresses to the Blockchain encrypted, select at least two couriers, and redirect shipments between different couriers without a trusted third party. Once the courier arrives at the customer's address, their schema utilizes a secure hashing to prove the delivery. However, their system increases complexity for buyers, who must select at least two delivery companies to ship their packages. Additionally, the proof of delivery is not linked to the package. Consequently, the courier can easily tamper with the asset to be delivered.

Hasan and Salah designed a blockchain-based solution for physical asset delivery (Hasan and Salah, 2018). Using automated ether payments, they analyzed a Proof of Delivery system to trade and track sold items between two parties. Each party deposits collateral equal to double the package price that each party risks losing if any behaves maliciously. Their protocol ensures asset handover verification through a key exchange the seller provides to the courier and the buyer. The solution also offers tamper-proof logs for auditability and traceability. However, the scheme does not fit a crowd-shipping environment since it requires a double package price deposit per order. Besides, the relationship between the key and the package is not stated.

Wang et al. presented an auditable protocol for transparent, tamperproof, and verifiable transactions between merchants, logistics companies, and consumers (Wang et al., 2019). The schema requires a regulator to authenticate users interested in participating in the network and register a smart contract with their data in the Blockchain. It has offered a pre-verification technique to prevent the parcel's replacement during delivery by the courier. Also, it discussed the return of the product in two cases; when consumers receive their products and when the delivery time of the products exceeds a predefined period. Like previous work, it requires a security deposit per order equal to the parcel's worth.

A novel cash-of-delivery solution was presented in (Ha et al., 2020) to address the courier's traceability during the delivery process and to integrate access control protocols to protect sellers' and customers' privacy. A hash code is created for each package based on its details and verified to ensure no change could occur to the order details. Smart contracts include a penalty mechanism when trouble arises, such as a damaged, missing, or incorrect package. The customer selects the shipper, and once agreed, the system sends the hash and the seller's contact to the shipper. Only the shipper mortgage a deposit equal to the package value, which is transferred to the seller in case of late delivery. When the customer receives the package, the shipper takes the cash payment and divides the profit with the seller.

These previous blockchain-based solutions adopt a collateral depositing approach to build trust between shipping parties. However, something else is needed to fit the crowd-shipping environment because it would be unfeasible for the courier to deposit collateral that equals each package value he will ship or double. The courier will have to deliver several parcels on the same route. A delivery driver typically ships 40–70 packages each day for close allocation of addresses and 125–200 packages if the addresses are super clustered (Wang, 2021). Hence, another mechanism is required to inject trust into the system while providing a practical and cost-effective solution.

# 3 HYPERLEDGER FABRIC

This section describes one of the most well-known distributed ledger frameworks, Hyperledger Fabric, and its transaction flow. HyperLedger Fabric was created by the HyperLedger blockchain open-source project and supported by the Linux Foundation. It aims to develop apps or solutions with a modular architecture and plug-and-play components for membership and consensus. It is a permissioned blockchain, and Table 1 lists its main components. There were several reasons for selecting Hyperledger fabric. Substantially, it is a non-cryptocurrency-based blockchain, meaning there is no PoW algorithm and crypto mining in Fabric. Thus, it delivers high scalability and fast transactions. Hyperledger Fabric is an open source with detailed documentation and several implementation examples. Furthermore, it supports the minimum technical requirements to build the proof of concept.

The ledger is divided into two sections, as illustrated in Figure 1, and a copy of ledger is maintained on each peer that joins the channel in Hyperledger Fabric. A blockchain data structure containing the blocks makes up the first part (of transactions). The second component, a world-state database, stores the most recent state following a block's commit. Upon successful validation, the peer commits the new block received from the ordering service into the ledger. The block is added to the blockchain, and each transaction is updated in the world-state. Most of the ledger is identical among peers inside a channel due to the consensus. However, Private Data is an exception, as only specific organizations store it in the world-state. In some cases, just a subset of the organizations requires to keep data private from other organizations on a shared channel. To meet this demand, Hyperledger Fabric introduces Private Data through the definition of data collection. All peers inside the subgroup can see the private data, while peers outside the subgroup will preserve a record of the private data hash as proof of data existence or for audit purposes. Each organization has an implicit data collection for private data by default.

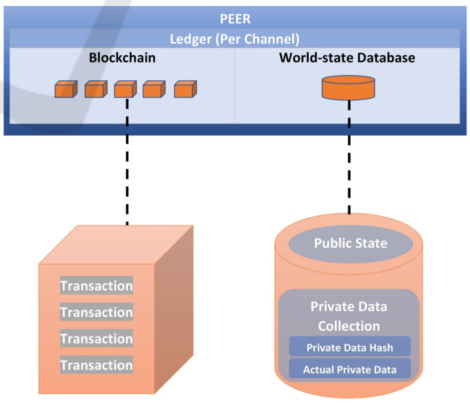


Figure 1: HyperLedger Fabric Data Structure.

Endorsement, Ordering, and Validation are the three stages of consensus in Hyperledger Fabric. Endorser nodes must endorse a transaction based on policy, such as (m out of n) signatures. The ordering phase accepts the approved transactions and consent to the order to be added to the ledger. Finally, the validation phase examines a block of arranged

Table 1: Major components of HyperLedger Fabric.

| Component | Description |
|---|---|
| MSP | Membership Services Provider (MSP) is implemented as a Certificate Authority to manage certificates used to manage and authenticate member identity and roles of all participants on the network. No unknown identities can transact in the Hyperledger Fabric network. |
| Peer | Peers are a vital element in the network that host ledgers and chaincode (smart contracts). An application interface, ledger data access, endorsement of transactions, and chaincode execution are all performed by a peer. Some peers can be endorsing peers which validate transaction requests from the client, commit the block received from the Orderer and update its ledger. |
| Orderer | Orderer are nodes which produces a block containing the endorsed transactions after sorting them according to the time they were received from peers, then distributes the blocks to all other peers. |
| Client | Clients act on behalf of the system end-user by submitting transaction-invocation requests to the endorsers and broadcasting transaction proposals to the orderers. |
| Chaincode | Chaincode refers to the smart contracts used by Hyperledger Fabric. Chaincode is a program that holds the system's business logic and executed when predefined conditions are met. When an application has to communicate with the ledger, the application invokes the Chaincode. Chaincode is deployed to all peers at the initialization stage of the fabric network. |
| Channel | Channels are a logical structure formed by multiple organizations to create a separate ledger of transactions. When configuring any channel, a set of policies must be agreed upon to govern the interactions between organizations and define the permission to invoke the chaincode deployed on this channel. |
| Organization | Organization is an entity that consists of a group of peers who have an identity (digital certificate) assigned by a Membership Service Provider. |

transactions to ensure accurate outcomes, including reviewing endorsement policy and double-spending. The current consensus algorithms in Hyperledger fabric are CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) to support different trust assumptions of a particular deployment or solution (Androulaki et al., 2018).
The Hyperledger Fabric transaction flow process consists of eight phases, as outlined below:

1. The user enrolls in the MSP through an application, and the MSP issues them a User ID and a certificate.
2. The user proposes a transaction to network peers.
3. Endorsing peers who received the transaction from the user perform a validation check on the client's identity to ensure they are authorized for their request. The transaction is then simulated using the pre-deployed Chaincode.
4. After successfully simulating the Chaincode, each peer gives the user their endorsement.
5. The user gathers peer endorsements and sends them to the orderer.
6. The orderer organizes the endorsed transactions received in the previous step in chronological order and constructs a block containing them.
7. Orderer distributes the block to all the network's peers after sorting the endorsed transactions received from peers.
8. Each peer updates its ledger by appending the new block to the prior block after receiving and verifying it. At this stage, the ledger is identical for all peers.

# 4 PROPOSED BLOCKCHAIN SOLUTION

This chapter demonstrates the design and implementation of a Blockchain Crowd-shipping platform that offers decentralization, rights to data accessibility, visibility, and transparency using blockchain features. The key objective is to provide a free-mediator platform that ships goods and valuables among parties without trust depending on a reputation model.

## 4.1 Operational Scenario

Parcel shipping processes include different actors with each assigned task and role. The seller, Customer, and Courier are the key players in the proposed system, with an organization for each actor in the blockchain network.

As illustrated in Figure 2, the shipping process scenario starts with creating the parcel the seller wants to ship to a customer. The parcel's private details, such as parcel size, quantity, appearance, price, etc., are stored in a private data collection and hashed to generate the parcel ID. The seller who owns the parcel ID can only access these details. Using the same function, the seller creates shared parcel data with the customer. Shared Data is stored in a private data collection called Parcel Collection, accessible by the seller and customer to track updates on the shipping process. The seller passes out of the band the parcel-ID and the parcel properties to the customer. Next, the customer adds the convenient shipping destination and time and verifies the mutual package properties before agreeing to transact. The provided parcel's private properties must generate a hash that identically matches the hashed parcel ID; otherwise, this step is failed. This step ensures that the agreed parcel information meets the customer's expectations.

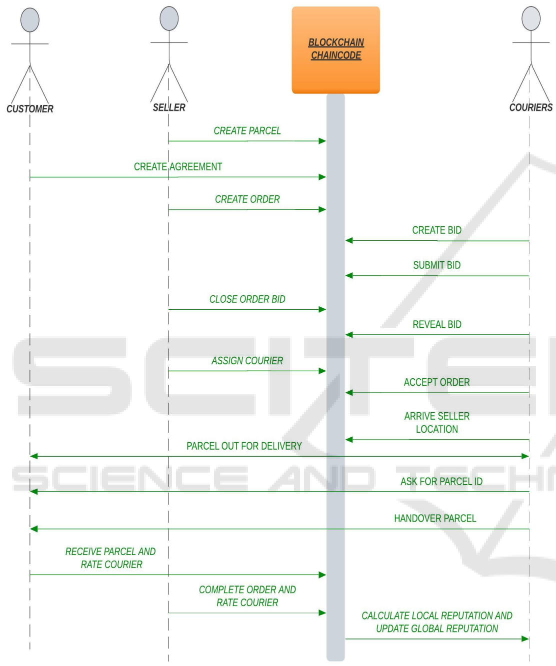


Figure 2: Sequence diagram of successful shipping process scenario.

Upon customer agreement, the seller creates an order to assign a courier to fulfill the shipping request. Any order consists of public data visible to any channel member. The order's public data includes the shipping date, time, locations, minimum reputation value of the courier, and maximum amount paid by the seller. On the other hand, the private data includes the assigned courier, shipping cost, parcel ID, and additional metadata. Order's private data is stored in a private collection shared between the selected courier and the seller.

Each order is created with an Open status to enable couriers to submit their bids (requested compensation). If the courier is interested in a particular order, first, he creates his Full bid in his organization's implicit private data collection. The Full bid includes the courier identity and requested pay. However, before adding the bid to his organization's implicit private data collection, the smart contract verifies that the courier's global reputation matches the minimum threshold specified in the order and the provided price is less than the maximum paid amount in the order. If the bid is created successfully, the courier submits the bid's hash to the order without revealing the cost.

Meanwhile, the courier's organization is added to the list of organizations necessary to approve order updates. After several bids join the order, the seller closes it to prevent additional bids from being submitted and allows couriers to reveal their bids. The smart contract automatically assigns the courier with the lowest pay if the seller and courier organizations endorse the same courier and pay. That prevents the seller from prematurely assigning a courier to the order or colluding with couriers with unfair prices. Next, the selected courier can accept or reject the order. If he rejects the order, a certain amount of his reputation will be deducted.

In the following stage, the seller updates the parcel and order state to Courier Assigned and transmits the exact pickup and drop-off locations out of the chain by email or any communication method to the courier. The courier arrives at the pickup location and notifies the seller on the chain by changing the order state to Courier Arrived. The courier picks up the parcel, and the seller updates the parcel and order state to Out For Delivery. When the courier reaches the customer's destination, the courier asks the customer for the parcel ID. If the parcel ID is correct, the courier will successfully update the parcel to Handedover to the customer. At the same time, the customer will update the parcel state to Received by Customer and provide an evaluation score for the courier. Finally, the seller can verify the correctness of the parcel and order states, provide an evaluation score of the courier service, and set the parcel as Delivered and the order as Completed. The smart contract automatically collects all required data to estimate and update the courier's reputation score.

The seller can cancel the order before the parcel is shipped. If the shipping order state change to courier arrived, the shipping process has started, and cancellation is not applicable after this point.

## 4.2 Proposed Reputation Model

Reputation plays a crucial role in establishing systemic trust since it represents the service's reliability and the behaviour of participating entities. Previous works in literature established trust by reserving collateral deposits from each entity as cryptocurrency. The use of collateral deposits is not practical in a decentralized crowd-shipping environment. Instead, this work constructs trust through a reputation-based network that uses blockchain immutability and offers a confidence reputation score in an environment that lacks a third party who maintains transparency and solves disputes between participants.

Each courier is assigned a reputation score that sellers will consider during selection. A high reputation score reflects good courier behaviour. Furthermore, unlike traditional crowd-shipping schemes where the reputation is centralized, managed, and controlled by a third party, our reputation system is entirely decentralized and implemented on the blockchain.

#### 4.2.1 Local Reputation

It is not fair to directly use user evaluations to determine the reputation scores of the courier. Multiple transactional elements such as shipping order time, shipping cost, the credibility of the evaluation's source, and the courier's completed orders are disregarded in the reputation's computation, making the reputation system vulnerable to attacks. To calculate the courier's reputation score, we weigh the received evaluation, denoted by $r$ and take the value of [-1,1], according to the following factors:

1. **Order Time:** If the courier accepts several shipping orders in a short time, the courier and the seller may collude to receive good ratings. Thus, for a current order, if the courier's previous order occurred a short time ago, the rating of the current order should be set as a relatively low value to deter the collusion attacks. Thus, the weighting factor of order time is defined using the Hyperbolic Tangent function in equation 1 as suggested by (Zhou et al., 2021):

$$\varphi(\triangle T) = tanh(\triangle T) \tag{1}$$

where $\triangle T$ refers to the time interval between the timestamp of current order $T$ and that of previous transaction $T'$ divided by $T_a$ which refers to the average frequency of user interaction in the system, $\triangle T > 0$ and $\varphi(\triangle T)$ ranges from 0 to 1. A distant time interval $\triangle T$ will lead to a higher value of the weighting factor.

2. **Shipping Cost:** It is not costly for the seller to create orders with low asked prices to boost courier ratings or unfairly submit low ratings to the courier. Therefore, to overcome this concern, the rating of an order should be related to the shipping cost amount. The weighting factor of the shipping cost amount is defined by equation 2, where $V$ refers to the shipping cost and $\psi(V) < 1$. The shipping cost is divided by 10 to avoid convergence to 1 very quickly:

$$\psi(V) = 1 - \frac{1}{(1 + \frac{V}{10})} \tag{2}$$

3. **Number of Shipping Orders:** A courier may increase his trust value by being active in the network and increasing the fulfilled shipping orders. To distinguish a courier with a high reputation for a few good orders from a courier with a high reputation but with a large volume of orders, we need to consider the number of transactions (Xiong and Liu, 2004). This factor is denoted by $TX_N$ and defined by considering the maximum number of completed orders by couriers in the network and assigning different weight values for different ranges. The network determines what factor's score to assign for each range of shipping requests and updates it periodically.

4. **Credibility of the Local Rating:** The proposed reputation model must be robust against unfair ratings. A participant may make false statements about the courier's service due to malicious motives. Consequently, a trustworthy courier may get many unsatisfactory ratings despite providing satisfactory service in every order. Therefore, we consider the fairness of the provided rating score for each order to protect the courier from such incidents. We adopt the approach presented in (Allahbakhsh et al., 2012). First, we calculate the average of all ratings given to a particular courier, say $c_j$, as per equation 3. $r_{kj}$ refers to the rating received by user $u_k$ toward the courier $c_j$. The $N_j$ refers to the total number of fulfilled shipping requests by courier $c_j$ .

$$\overline{R_j} = \frac{\sum_{k \in N_j} r_{kj}}{N_j} \tag{3}$$

In the second step, we calculate the average of all ratings given to courier $c_j$ by the rater, say $u_i$, using equation 4. The $N_{ij}$ refers to the number of ratings that $u_i$ has provided toward the courier $c_j$.

$$\overline{R_{ij}} = \frac{\sum r_{ij}}{N_{ij}} \tag{4}$$

In the third step, we calculate the standard deviation of all ratings given to a courier $c_j$ as shown in equation 5.

$$SD_j = \frac{\sqrt{\sum_{k \in N_j} (r_{kj} - \overline{R_j})^2}}{N_j} \tag{5}$$

In the last step, we measure the rating credibility factor as follows:

$$Cr_{ij} = \begin{cases} \frac{\overline{R_j} - SD_j - \overline{R_{ij}}}{Max_{Rep}} & \text{if} \quad \overline{R_{ij}} < (\overline{R_j} - SD_j) \\ 1 & \text{if} \quad (\overline{R_j} - SD_j) \leq \overline{R_{ij}} \\ 1 & \text{if} \quad (\overline{R_j} + SD_j) \geq \overline{R_{ij}} \\ \frac{\overline{R_{ij}} - (\overline{R_j} + SD_j)}{Max_{Rep}} & \text{if} \quad (\overline{R_j} + SD_j) < \overline{R_{ij}} \end{cases} \tag{6}$$

where $Max_{Rep}$ refers to the maximum value of the reputation score and is equal 1.

According to Equation 6, the averages falling in $\overline{R_j} \pm SD_j$ are trustworthy and dependable. Still, those that fall out of that range have very low credibility, and their impact on reputation is decreased. The $Cr_{ij}$ shows how close is the judgment of $u_i$ to the majority consensus about courier $c_j$'s trustworthiness. Thus, we use credibility to reduce the effect of ratings from raters who disagree with the majority consensus about the courier. It is worth noting that the credibility factor could be difficult to measure if both rater and the courier are transacting for the first time. Therefore, the credibility factor value will be 0.5 as the probability of a fair/unfair rating score is 50%.

Finally, for each shipping request, we weigh the received rating $r$ (positive or negative) by all factors using equation 7. Local reputation score ranges from 0 to 1.

$$e = \frac{(r \times Cr) + \psi(V) + \varphi(\triangle T) + TX_N}{4} \tag{7}$$

### 4.2.2 Global Reputation

As mentioned before, each courier has a local and global reputation score. When the courier joins the network, an initial global reputation score is assigned, equaling 0.5. The global reputation increases or decreases once a new local score is added. Also, it decays if the courier has not been active after some time. We have adopted the suggested model by (Truong et al., 2021). Local reputation $e$ could be satisfactory or unsatisfactory depending on a predefined threshold $\theta$. The amount of increase, decrease, and decay depends on the local reputation score $e$ and the current value of the global reputation $GRep$ , which can be modelled by linear difference equations and a decay function as follow:

**Increase Model.** The current global reputation score denoted by $GRep$ increases once updated with a satisfactory transaction (at the time t, indicated by the local reputation score $e_t \geq \theta$) that follows the linear difference equation:

$$GRep_t = GRep_{t-1} + e_t \times \Delta GRep_t \tag{8}$$

where $\Delta GRep_t = \alpha \times (1 - \frac{GRep_{t-1}}{Max_{Rep}})$ and $\alpha$ is the maximum increase value of global reputation score in two consecutive shipping requests and $Max_{Rep}$ is the maximum global reputation and equals 1.

**Decrease Model.** Similarly, $GRep$ decreases if the local rating is unsatisfactory (indicated by the local reputation score $e_t \leq \theta$), following the equation:

$$\begin{aligned} GRep_t = Max(&Min_{Rep}, \\ &GRep_{t-1} - \beta \times (1 - e_t) \times \Delta GRep_t) \end{aligned} \tag{9}$$

The $Min_{Rep}$ is the minimum global reputation and equal 0. The decrease rate $\beta > 1$ implies that it is easier to lose the global reputation value due to an unsatisfactory rating than to gain it (a satisfactory rating).

**Decay Model.** Global reputation decays if there is no transaction after a period of time. The decay rate is assumed to be inversely proportional to the strength of the trust relationship of the courier (value of the Reputation). Based on these observations, the Decay model is proposed as follows:

$$GRep_t = Max(Min_{Rep}, GRep_{t-1} - \Delta Decay_t) \tag{10}$$

where $\Delta Decay_t = \delta \times (1 + \gamma - \frac{GRep_{t-2}}{Max_{Rep}})$ , and $\delta$ is the minimum decay value ensuring any global reputation degenerates if it is not maintained. And $\gamma$ is a decay rate controlling the amount of the decay. The network administrator can identify the inactivity threshold (e.g., three months) and periodically compare it to the couriers' last shipping request. If the courier has been inactive more than the threshold, his global reputation declines using equation 10.

## 5 PROOF OF CONCEPT IMPLEMENTATION

As proof of concept, we implement our decentralized crowd-shipping last-mile delivery framework using Hyperledger Fabric. The source code of the proposed framework is publicly available on GitHub (WASPLab, 2022). In this section, we briefly describe the blockchain network and how the network's different nodes interact. Besides, the deployment and implementation details.

## 5.1 Proposed Blockchain Network Model

The blockchain network consists of four organizations, with one peer for each, as depicted in Figure 3. A dedicated certificate authority is assigned to each organization. The Peer node is intended to be an endorsing peer where the system's Chaincode resides. Each peer maintains a current state database as the couch DB. The sequence of our proposed work is creating a channel; each peer must join the channel, install the Chaincode, and approve it. If the peer receives sufficient approvals from the organizations, it commits the Chaincode, invokes it, queries it, and enables client communication with Postman API.

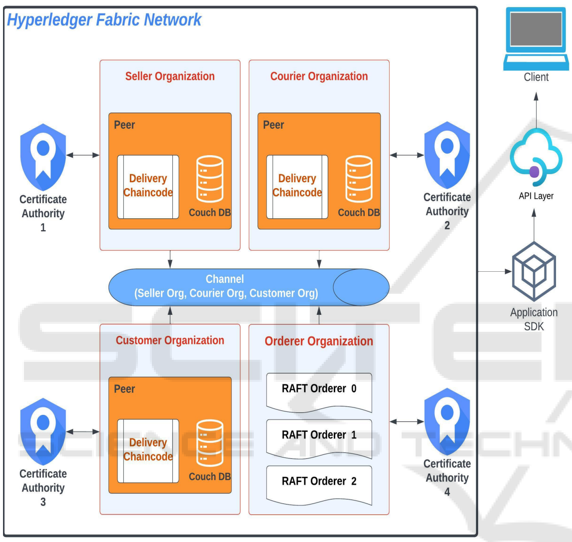


Figure 3: Proposed LMD Platform Network Architecture.

Each actor in the system has a specific organization that will provide the entity with the relevant authorization and Chaincode access. The certificate authority is used to manage and authenticate member identity and roles of all participants on the network. No unknown identities can transact in the Hyperledger Fabric network. Each organization has a peer node that carries out the application interface, ledger data access, endorsement of transactions, and Chaincode execution. The Oderer organization consists of three Orderer nodes that produce a block containing the endorsed transactions after sorting them according to the time they were received from peers. Then, they distribute the blocks to the leader peer in the network, which then forwards the blocks to other peers. The peers receiving the data will use gossip protocol to disseminate the data to all peers of the same channel.

Hyperledger Fabric offers several implementations for achieving consensus between ordering service nodes, Raft, Kafka, and Solo. In our network, we use Raft, a crash fault-tolerant (CFT) that uses the ”leader and follower” architecture, each channel elects a leader node, and the followers replicate that node's decisions. Due to its endorsement policy of majority vote, Raft provides a means for high availability for ordering services.

The platform's business logic comprises Chaincode, application SDK, and Application Programming Interface (API) testing tool, which works together to deliver the application's features. Chaincode consists of the Data model designed to define the data structures necessary for the application network (Chaincode and Application SDK). The main smart contract is written in Golang and verifies invoker roles, and executes the associated transaction functions for each service capability's logic.

The application SDK provides access to Chaincode running within that network and to which transactions can be submitted or queries can be evaluated. It is written in javascript and uses the Gateway class as the network entry point. For testing Application Programming Interface (API), we use the Postman tool that enables the client to interact efficiently with the system and determine how system resources are defined and addressed.



## 5.2 Development Environment

One Virtual machine running Ubuntu Linux 20.04 has been deployed on a 64-bit machine with 16 GB of RAM and 2.3 GHz Intel Core i7 CPU. The requirements and specification of our proposed network has been shown in Table 2. The following steps are taken in order to run the blockchain network:

1. Generate Crypto Materials for Seller, Customer, and Courier Organizations and RAFT Orderer. It creates the node organization unit materials related to CA, MSP, peers, TLSca, admins and users for all organizations.
2. Create Channel Artifacts auch as genesis block and channel transaction files, and anchor peers.
3. Creating and Joining Channel.
4. Delivery Chaincode Deployment.
5. Install, Approve, Commit, and Invoke Delivery Chaincode.
6. Launch the postman API server in order to allow the interaction with the application and the Hyperledger Fabric's local environment.

Table 2: Requirements and specification of proposed LMD Blockchain network.

| Requirements | Specification |
|---|---|
| cURL Tool | 7.68.0 |
| Docker engine | 20.10.18 |
| Docker Composer | 1.25 |
| Go | 1.14.1 |
| Node JS | 13.14.0 |
| NPM | 6.14.4 |
| Hyperledger Fabric | 2.1.1 |
| VS Code | 1.74.1 |
| Postman API | 9.31.25 |
| Hyperledger Caliper | 0.5.0 |
| Couch DB | 0.4.20 |
| Certificate Authority | 1.4.7 |

# 6 DISCUSSION AND RESULTS

This section presents a feature-based comparison between our proposed system and related work reviewed in section 2. Security, privacy, and scalability aspects are also discussed. Then, we conduct reputation test scenarios to evaluate the effectiveness of our reputation model, finally assess the application performance using Hyperledger Caliper, and discuss the obtained results.

## 6.1 Security Analysis

The suggested LMD solution leverages key security features from blockchain by design, such as decentralized trust, integrity, non-repudiation, and availability. Although existing blockchain and smart contract technologies still have performance and security threats, we assume that the decentralized feature of the BC makes it impossible for an adversary to compromise the blockchain network and alter the ledgers' contents.

Each actor in the system has a digital identity encapsulated in an X.509 digital certificate issued by a Certificate Authority (CA). These identities determine the permissions over resources and access to information. Also, by design, our framework is secured against Man-In-The-Middle (MITM) attacks and replay attacks. Every message exchange is cryptographically signed and timestamped, ensuring nobody can repudiate their activities later. Integrity is essential in preventing critical information tampering. The proposed framework provides the ability to use transaction logs to track back historical occurrences.

In our system, a seller and customer must perform a complete shipping request with the courier to be able to provide a rating. The suggested reputation strategy itself can thwart several reputational attacks. For instance, Whitewashing is mitigated because a central authority registers participants to the permissioned network. A participant can only rejoin with the network administrator's permission if their identity is revoked. Further, since registration requires a form of identification, Sybil's attacks are prevented.

Table 3 compares the examination of several features offered in previously explored works in section 2 to the proposed system. We examine each system using standard security criteria and offered features.

## 6.2 Reputation Model Testing Results

This section evaluates the reputation model's effectiveness in four different scenarios. Each scenario examines the significance of each factor utilized in the model. Before demonstrating the scenarios and their results, it is worth noting that the parameters controlling the Reputation model ($\alpha, \beta, \gamma, \delta$) must be optimized per the business requirements due to its influence on the global reputation score growth or dwindling speed.

Our data is from a unique dataset of ride-hailing journeys produced by RideAustin, a nonprofit ridesharing service based in Austin, Texas (Austin, 2017). The dataset has several features; we only select Ride ID, Rider ID, Driver ID, Rating, Order Creation Time, and Ride Cost. Table 4 shows the parameters configuration used in the following experiments to assess our proposed reputation model.

The first test case examines the reputation model resilience to the malicious behavior of the courier if he conspires with a seller and performs consecutive requests to boost his reputation score. Figure 4 depicts two couriers with identical data. The difference is that the malicious courier fulfilled requests at intervals of less than an hour. In contrast, the non-malicious courier has more than one hour difference between requests. The model can detect this behaviour and control the courier's reputation. The non-malicious courier's global reputation gradually increased and reached 0.98, while the malicious remained between 0.50 - 0.55.

The second test case analyzes the reputation model action toward unfair ratings. Figure 5 presents two identical couriers who accomplished 50 requests with a positive rating rate of 70% and 30% negative ratings. The first reputation is computed using our proposed computation, while the other is without the credibility factor. After fifty shipping requests, the attained reputation for the first and the second courier reached 0.83 and 0.78, respectively. We notice that

Table 3: Features Comparison Between Related Work and The Proposed Solution.

| System Name | Accountability | Auditability | Anonymity | Courier Reputation | Proof of Delivery | Traceability | Scalability |
|---|---|---|---|---|---|---|---|
| Lelantos | X | X | ✓ | X | ✓ | X | X |
| Single and Multiple Transporters | ✓ | ✓ | X | X | ✓ | ✓ | X |
| Auditable Protocols for Fair Payment | ✓ | ✓ | Only for Customers | X | ✓ | ✓ | X |
| DeM-CoD | ✓ | ✓ | X | X | ✓ | ✓ | X |
| Proposed System | ✓ | ✓ | X | ✓ | ✓ | ✓ | ✓ |

Table 4: Reputation Test configuration.

| Parameter | Value |
|---|---|
| 1-25 requests<br>26-50 requests<br>51-75 requests<br>$>$ 76 requests | 0.25<br>0.50<br>0.75<br>1 |
| Normal time difference between two consecutive requests $T_a$ | 1 hour |
| Local reputation threshold $\theta$ | 0.5 |
| Maximum increase value $\alpha$ | 0.1 |
| Decrease rate $\beta$ | 1.6 |

using the credibility factor alleviates the negative rating impacts provided unfairly by the same user.

The third test case investigates the influence of shipping cost over reputation. Figure 6 depicts two couriers with identical data except for the shipping cost. All the shipping costs of the malicious courier's requests are set to 3$, while the non-malicious courier to 10$. The cause of the steady reputation of the malicious courier for the first 25 requests is that the local reputation score is less than the threshold that distinguishes between satisfactory and unsatisfactory requests. When the malicious courier completed more than 25 requests, the number of transactions factor became 0.50, which made the local reputation score higher than the specified threshold. Later, the courier's reputation gradually increases, similar to any honest courier, which indicates that detecting low-priced requests and malicious behaviour using the model is temporal. Once the courier has a higher weight of the other factors, it would be undetectable.

The fourth test shows how the reputation grows with increasing the total number of shipping requests. Figure 7 presents two couriers; the first completed 50 shipping requests while the other 25 shipping requests. The first courier's reputation value is 0.98, while the other is 0.89. The weight of the number of shipping requests factor will increase as long as the courier fulfills more requests; consequently, his global reputation will thrive.

### 6.3 Performance Evaluation

Blockchain systems utilize a performance measuring tool called Hyperledger Caliper (HL Caliper) (Hyperledger, 2018) to evaluate the system based on several performance indicators. This study conducts experiments locally using the Hyperledger Caliper tool to assess the transaction latency and throughput. We perform two experiments using the Fixed Rate Controller. This controller submits the transactions at a predetermined interval, denoted as TPS (transactions per second). We evaluate the system performance for each experimental scenario under two different send TPS rates; at 20 TPS and 40 TPS. Five different network loads (total Transactions) are selected to investigate the TPS rate impact on the platform's network performance in terms of throughput and latency.

Figures 8 and 9 illustrate the transaction throughput with TPS equals 20 and 40, respectively. It is observed that the throughput at a transaction send rate of 40 is slightly higher than when it is 20. Also, when the TPS is 20, it shows consistency in the throughput, which reflects the reliability and availability of Hyperledger. The throughput ranges between 17-19, except for the Create Order function. Create Order function has less throughput since it performs one read operation and two write operations (one on the ledger and the other on private data collection). When the TPS is 40, the throughput is approximately similar for each function in figure 9, disregarding the transaction load.

Figures 10 and 11 describe the latency after executing the Chaincode's functions, using 100 to 500 simultaneous transactions. it is noticed that when TPS is 20, the average latency follows a particular pattern and remains consistent. The Create Order achieved the highest average latency, 9.33 seconds when TPS is 20 and 16.43 seconds when TPS is 40. Create Order transactions require more time to be executed and written successfully on world state and private data collection. Furthermore, there is a continuous growth in the average latency as the number of transactions increases when TPS is 40. The reason is that the transactions waiting at the orderer node are considerably growing each second.

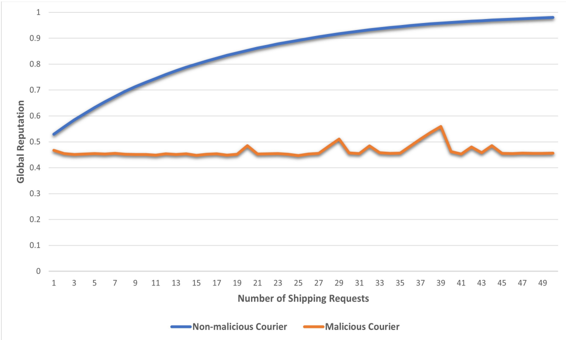


Figure 4: Test Case 1 - Timestamp Factor Impact.


Figure 5: Test Case 2 - Credibility Factor Impact.


Figure 6: Test Case 3 - Shipping Cost Factor Impact.

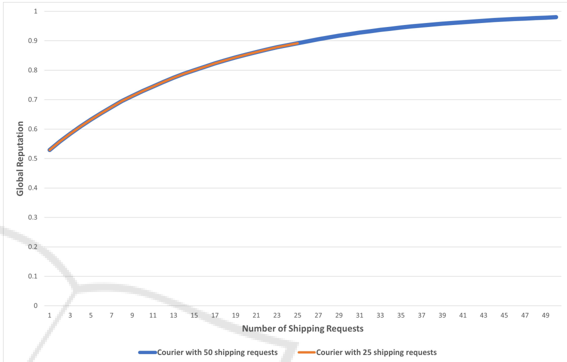


Figure 7: Test Case 4 - Number of Shipping Requests Factor Impact.

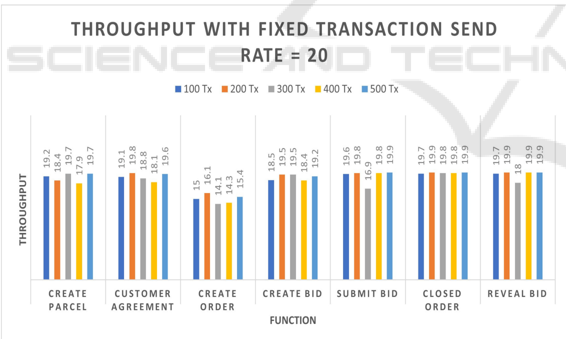


Figure 8: Experiment I - Throughput with TPS = 20.

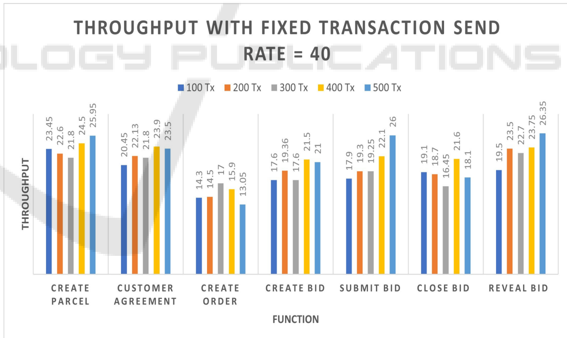


Figure 9: Experiment II - Throughput with TPS = 40.

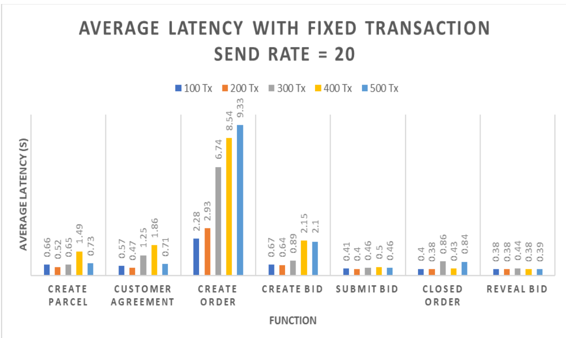


Figure 10: Experiment I - Latency with TPS = 20.

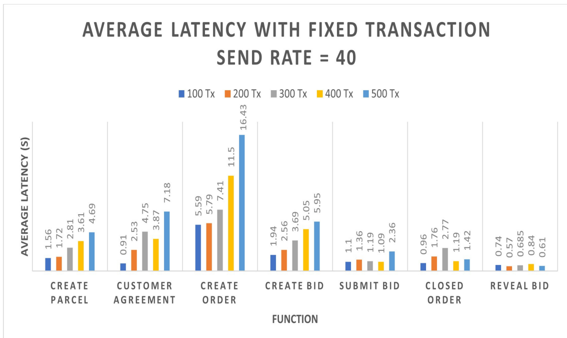


Figure 11: Experiment II - Latency with TPS = 40.

## 7 CONCLUSIONS

This paper proposes a blockchain-based crowdshipping system that eliminates the need for third-party involvement, using a reputation management algorithm to encourage honest conduct and aid sellers in selecting trustworthy couriers. The study includes a prototype implementation of the proposed platform and conducts performance measurements, experiments, and feature comparisons to related research. The proposed schema uses Hyperledger Fabric, which is entirely free and open-source and avoids costly consensus protocols to develop a feasible and practical solution. However, the research has limitations, such as the need for sellers to wait for couriers to reveal their bids and the potential need for off-chain computation of couriers' reputations in large-scale environments. Future work could also consider allowing couriers to submit multiple bids and deploying the application on a Cloud with high computational resource capabilities.

## ACKNOWLEDGEMENTS

This project was partly supported by collaborative research funding from the National Research Council of Canada's Artificial Intelligence for Logistics Program.



## REFERENCES

Allahbakhsh, M., Ignjatovic, A., Benatallah, B., Beheshti, S.-M.-R., Foo, N., and Bertino, E. (2012). An analytic approach to people evaluation in crowdsourcing systems. https://arxiv.org/abs/1211.3200.

AlTawy, R., ElSheikh, M., Youssef, A. M., and Gong, G. (2017). Lelantos: A blockchain-based anonymous physical delivery system. *2017 15th Annual Conference on Privacy, Security and Trust (PST)*.

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A. D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K. A., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S. W., and Yellick, J. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. *Proceedings of the Thirteenth EuroSys Conference*.

Austin, R. (2017). Ride-austin-june6-april13 - dataset by ride-austin. https://data.world/ride-austin/ride-austin-june-6-april-13.

Breen, K. (2019). Food delivery app doordash reports data breach affecting 4.9m users - national. https://globalnews.ca/news/5957123/doordash-data-breach/.

Dolan, S. (2023). The challenges of last mile delivery logistics and the tech solutions cutting costs in the final mile. https://www.insiderintelligence.com/insights/last-mile-delivery-shipping-explained.

Evans, P. (2020). Food delivery apps cut some restaurant fees amid surging demand due to covid-19 — cbc news. https://www.cbc.ca/news/business/food-delivery-apps-fees-1.5765790.

Gatta, V., Marcucci, E., Nigro, M., Patella, S., and Serafini, S. (2018). Public transport-based crowdshipping for sustainable city logistics: Assessing economic and environmental impacts. *Sustainability*, 11:145.

Ha, X. S., Le, H. T., Metoui, N., and Duong-Trung, N. (2020). Dem-cod: Novel access-control-based cash on delivery mechanism for decentralized marketplace. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 71–78.

Hasan, H. R. and Salah, K. (2018). Blockchain-based solution for proof of delivery of physical assets. *Lecture Notes in Computer Science*, page 139–152.

Hyperledger (2018). Measuring blockchain performance with hyperledger caliper. https://www.hyperledger.org/blog/2018/03/19/measuring-blockchain-performance-with-hyperledger-caliper.

Le, C. (2022). Canada - ecommerce. https://www.trade.gov/country-commercial-guides/canada-ecommerce.

N, B. (2020). Hackers leaked ubereats data on darkweb. https://cybersecuritynews.com/hackers-leaked-ubereats-data-on-darkweb/.

Placek, M. (2022). Couriers and local delivery services market size in canada 2018-2021. https://www.statista.com/statistics/1156206/couriers-and-local-delivery-services-market-size-canada/.

Truong, N., Lee, G. M., Sun, K., Guitton, F., and Guo, Y. (2021). A blockchain-based trust system for decentralised applications: When trustless needs trust. https://arxiv.org/abs/2101.10920.

Wang, M. (2021). I tried delivering parcels for a day. https://www.linkedin.com/pulse/i-tried-delivering-parcels-day-mark-wang.

Wang, S., Tang, X., Zhang, Y., and Chen, J. (2019). Auditable protocols for fair payment and physical asset delivery based on smart contracts. *IEEE Access*, 7:109439–109453.

WASPLab (2022). Wasplab/bc_crowdshipping: This is the blockchain network of a crowdshipping platform using hyperledger fabric. https://github.com/WASPLab/BC_Crowdshipping.

Xiong, L. and Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(07):843–857.

Zhou, Z., Wang, M., Yang, C.-N., Fu, Z., Sun, X., and Wu, Q. J. (2021). Blockchain-based decentralized reputation system in e-commerce environment. *Future Gener. Comput. Syst.*, 124(C):155–167.