2017 IEEE 37th International Conference on Distributed Computing Systems Workshops

# On the integration of event-based and transaction-based architectures for supply chains

Zhijie Li, Haoyan Wu,
Brian King and Zina Ben Miled
*Electrical and Computer Engineering Department*
*Purdue School of Engineering and Technology, IUPUI*
*Indianapolis, IN*
*Email: lizhij@iupui.edu, haoywu@umail.iu.edu,*
*briking@iupui.edu, zmiled@iupui.edu*

John Wassick,
Jeffrey Tazelaar
*The Dow Chemical Company*
*Midland, Michigan, USA*
*Email: JMWassick@dow.com,*
*JRTazelaar@dow.com*

***Abstract*—Affordable and reliable supply chain visibility is becoming increasingly important as the complexity of the network underlying supply chains is becoming orders of magnitudes higher compared to a decade ago. Moreover, this increase in complexity is starting to reflect on the cost of goods and their availability to the consumers. This paper addresses two key issues in the distribution phase of the supply chain, namely, affordability and pseudo real-time visibility of truckload activities. The proposed framework creates a digital thread that tracks the pseudo real-time status of the shipment making the physical distribution process completely transparent to the stakeholders. The architecture of the framework is based on a dynamic hybrid peer-to-peer network and a private/public blockchain data model that leverages emergent sensor technologies.**

## 1. Introduction

In the past three decades, the digitization of information and its exchange went through several phases that were driven by demands and technological innovations. Going from paper-based records within an institution to isolated applications that cover one or more aspects of the business was an initial phase. During this phase, multiple applications and underlying databases were used for accounting, human resources, maintenance, etc. This phase was followed by a trend towards enterprise resource planning (ERP) systems that integrated the digital thread across various activities within each institution including accounting, human resources management, payroll, maintenance, warehousing and logistics with varying emphasis depending on the core business of the organization. Additional advances and demands led to the globalization of these ERP systems especially for multi-nationals and corporate groups. Moreover, the digital thread started to extend to cover the external relations of the organization with its customers (e.g., customer portals and customer relations management systems - CRMs) and its partners (e.g., suppliers, financial institutions, service providers). This latter extension was facilitated by the emergence of the cloud computing business model which allows different partners to exchange information using a common platform. These platforms are nowadays seamlessly integrated within the business processes internal to an organization. Compared to their predecessors, these new systems offer significant improvements in efficiency and transparency. However, with the recent advances and ubiquity of sensor technology and IOT, there are increasing demands for higher efficiency and transparency.

This paper focuses on addressing the issue of real-time transparency in the physical distribution phase of the supply chain and proposes a digital model that integrates a distributed event-based system with the traditional transaction-based system.

Supply chain consists of a complex and dynamic set of interactions and trade-offs between suppliers, manufacturing, warehousing, carriers, and customers to deliver the right product, at the right time, and in the right condition [1]. Current information systems and standards that support supply chain include the Electronic Data Interchange (EDI) [2], an ANSI X.12 standard, and EDIFACT [3] a similar standard used in Europe. These standards digitally reference and capture the content of various documents used in the supply chain including invoices, purchase order, bill of lading as well as messages such as the carrier shipment status message (i.e., EDI 214). The carrier shipment status message is of particular interest in this paper because it is used to communicate the real-time status of the shipment in the proposed architecture. Adopting EDI as a baseline facilitates the interoperability of the proposed framework with existing supply chain information systems. The EDI 214 document includes the shipping source, the shipping location, pickup and estimated delivery times and the shipment status details.

Both the EDI and the EDIFACT standards are used within current Supply Chain Operating Networks (SCONs) [4], [5] which enable the exchange of electronic documents across trading partners based on the movement of goods. SCONs are cloud-based solutions and often seamlessly integrate with the individual partners' ERPs. However, the value derived and the success of SCONs are directly proportional

2332-5666/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCSW.2017.51

to the level of participation by a given industry sector or trading group. Furthermore, the competitiveness of a given company is directly related to its ability to venture into new industries and sectors which today requires the capability to support multiple SCONs, often leading to inefficiencies and high costs [6]. As a result of these limitations, there is a wide consensus in the supply chain community that the current transactional layer provided through multiple centralized SCONs is not sufficient to support the recurrent requirements for supply chain visibility [7]. The model proposed in this paper aims at addressing these issues by using

- shipment-centric peer-to-peer (P2P) sub-networks that maintain shipment information using a private ledger and
- a centralized semi-public ledger that indirectly maintains shipment location. Entries in this ledger are validated by external monitors.

Section 2 of this paper describes the proposed architecture. Section 3 discusses the format used in the data exchanges and its relation to the currently used data exchange standards. Section 4 introduces the blockchain data model designed for the proposed framework. Section 5 reviews related work. Section 6 summarizes the contributions of this paper and plans for future work.

## 2. System Overview

The architecture of the proposed hybrid P2P physical distribution (HP3D) framework consists of a collection of dynamic P2P sub-networks. These sub-networks are created on demand, emulate the end-to-end movement of the shipment and terminate when the delivery of goods is completed. Each sub-network allows stakeholders to share information related to a given shipment and provides them with real-time visibility in the physical distribution segment of the supply chain. There are three types of components in HP3D: index server, peers and administrative nodes.

### 2.1. Index Server

The index sever dynamically collects and shares the IP addresses of the peers. It does not participate in the data exchanges between peers. In fact, the index server mainly handles two types of requests: client query requests and client update requests. The former type of requests is an query issued by a peer about the IP address of a target client ID. The latter consists of a heartbeat message sent by the peer to update its dynamic IP address within the server.

The light-weighted index server consists of a database for storage of peer-related information and an application layer that handles requests from the peers. The application layer is implemented in Golang [8] and uses JSON as the data exchange format. Each data exchange includes four fields, namely, request type, client name, client ID and client IP address. The index server uses the JSON format to communicate with peers as well as with the back-end database which is implemented in MongoDB [9]. The structure of the exchange is defined as:

```
type indexServer struct{
    ReqType int
    timestamp time
    IpAddr string
    ID bson.ObjectId `bson:
    ``_id,omitempty'''
}
```

where ReqType is an integer value that represents the request type. As mentioned earlier, there are two types of requests, client query and client update which are encoded as 0 and 1, respectively. The timestamp field represents the time of the latest update (i.e., heartbeat message). IPAddr is the IP address of a peer. The ID field is a unique hexadecimal number that is automatically generated by MongoDB for each peer when it initially registers with the index server. When the index server receives a type 0 request (i.e., client query), it searches through the database for a record with the same ID field value and reply with the corresponding IP address in the record. A 0.0.0.0 will be returned if the ID does not exist and a 1.1.1.1 is returned when the target peer is not active. A peer will be tagged inactive when the timestamp is older than the current system time by a predefined threshold.

### 2.2. Peers

A given peer may assume more than one role with respect to different shipment transfers. For instance, a carrier in one transfer can also be a customer in another transfer. The peers are also classified into two categories based on their platform: mobile or X86. However, all peers have the same software architecture regardless of their role or platform. In general, a peer is expected to have limited resources (e.g., a mobile device) and its application consists of three tiers: a presentation layer, a middle layer and a local database which is flushed regularly.

**2.2.1. X86 Peers.** The data flow across the three layers of the X86 peer application is shown in Figure 1. The presentation tier consists of a local HTTP server responsible for processing user requests and invoking the appropriate functions. It is implemented using HTML, CSS, JavaScript and Golang.

The middle layer of an X86 peer application is implemented using Golang and MGO. It is responsible for the execution of the following processes:

- Handling incoming user requests from the presentation tier
- Receiving signals from the sensor gateway
- Generating new events and updating the local database
- Sending updated shipment information to other stakeholders.

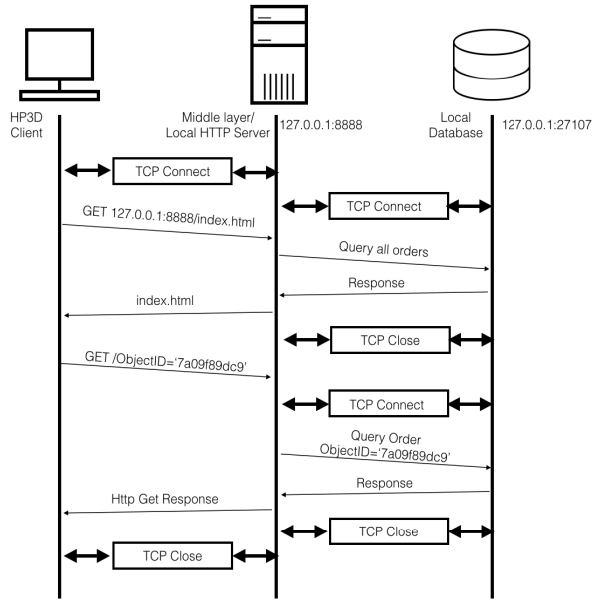


Figure 1. Data flow across the presentation, middle and database layers of the peer application

Handling user requests from the presentation tier is performed by using Golang socket programing. A user request is translated to a MongoDB query that retrieves one or more shipment information from the database. The request consists of an HTTP GET method invocation sent from the presentation tier to the middle tier. Once the request reaches the middle tier, it will trigger the corresponding database query. The query is processed by using the MongoDB driver MGO. The result is returned to the presentation tier through an HTTP PUT method.

The middle tier is also responsible for processing an incoming sensor signal and for sending updated shipment information. Once a sensor signal is received, the middle tier will update the local database using MGO and will send the updated information in JSON format to the other peers that are participating in the shipment-centric network (Figure 2). When a peer receives updated shipment information, it will in turn update its own local database by using MGO.

The events handled by the middle layer can be classified into three main types - general shipment information updates, geolocation updates and manual confirmation updates. A general shipment information update is issued by one of the stakeholders when a sensor signal is received (e.g., from a bar code scanner or RFID signal [10]). A geolocation update is specific to a GPS sensor signal. Confirmation updates are the third type of events and consist of updates that involve manual user inputs. An example of a confirmation update is when a customer manually confirms that the goods have been delivered. All of these events are stored in the database and create a distributed private blockchain [11] sub-ledger.

The third tier in the proposed HP3D is the data management tier. It is implemented in MongoDB in the case of X86 peers. MongoDB is a noSQL database management system. The database consists of a set of collections where each collection holds a different type of documents. The number of attributes and their corresponding data size can vary from one document to another. This feature, in addition to the fact that MongoDB also simplifies query design and processing made it the database management of choice for the proposed framework. The structure of MongoDB documents is also flexible as it allows various levels of nesting while still permitting easy data access to each document. Furthermore, fields associated with each shipment can be consolidated into a single document instead of potentially being scattered across multiple tables in a relational database.

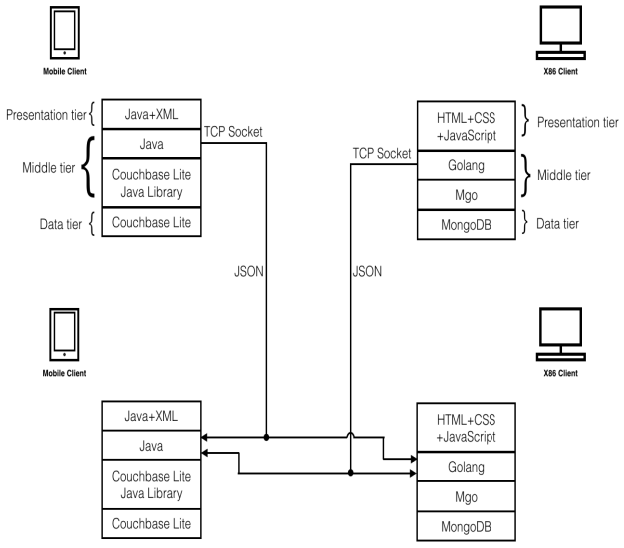


Figure 2. P2P Communication Flow

**2.2.2. Mobile Peer.** The mobile peer has the same three-tier architecture as the X86 peer (Figure 2). The presentation tier and the middle tier are developed by using JAVA, Android studio and XML. Furthermore, Couchbase lite [12], a noSQL database similar to MongoDB, is used for the data management tier. A Couchbase Lite database consists of set of buckets where each bucket stores a list of documents. The JAVA application accesses the database through the Couchbase lite driver.

Most of the functionalities of the mobile peer are the same as the X86 peer. The user interacts with the application via the presentation tier. The middle tier is responsible for sending/receiving shipment information updates to/from other peers as well as for updating its local database. However, the mobile device application needs additional functionalities since it is also used to acquire information from the field.

For example, when the supplier has transferred the shipment to the loading area for pick-up by the carrier, the mo-

bile peer application will scan the shipment information by using, for instance, a QR code scanner. Once the information is read, the mobile peer will send the updated information to all the peers that are involved in the shipment. In addition, the mobile peer is designed to acquire GPS information in order to report the geolocation of the truck and associated shipment. The geolocation is the longitude and the latitude generated by the GPS sensor in a mobile device and is represented by using the EDI 214 [13] standard as discussed in Section 3.

### 2.3. Administrative Node

A special node called the administrative node is also needed in the proposed framework. Each partner will have a designated administrative node. This node interacts with the enterprise resource planning system (ERP) of the partner and is able to retrieve order information and warehouse sensor information. In addition, it participates in all the shipment-centric sub-networks associated with a given partner. As such, it is able to act as a persistent record for all messages related to shipments involving the corresponding partner.

The administrative node has the same three-tier architecture as the other peers. The presentation tier is an HTTP server, the middle tier is implemented by using Golang and the data management tier is implemented by using MongoDB. All the messages that the administrative node receives are in JSON format and can be stored directly into the MongoDB database without any additional parsing.

One of the main motivation for the administrative node is its function as a data source when active peers drop out of the sub-network. For instance, if a mobile peer experiences a loss of connectivity, once it reconnects, it can retrieve the most recent shipment updates from the administrative node in its organization.

## 3. Data Exchange

In a supply chain information system, the data exchange template is complex and may include information about capital flow, contractual obligations and payment terms. The focus of this paper is on the transport phase of the supply chain. Therefore, our aim is to exemplify the sharing of field information among trading partners. For illustration purposes, we also limit the trading partners to the simple case of supplier, customer and carrier. In practice, several other parties are involved including, for example, manufacturers, assemblers, distributors, freight forwarders, brokers, financial institutions, etc. Furthermore, we only focus on the activity of truckload transportation.

A root struct called EDI214 which adheres to the EDI standard is created in Golang as shown below.

```
type EDI214 struct{
    EnvelopDet EnvelopDetail
    TCSSM TCSSMessage
    EnvelopSum EnvelopSummary
}
```

```
type TCSSMessage struct{
    THeading TCCSSMHeading
    TDetail TCCSSMDetail
    TSummary TCCSSMSummary
}
type TCCSSMDetail struct{
    L1000 []Loop1000
    L1100 []Loop1100
    L1200 []Loop1200
    L1300 []Loop1300
}
type Loop1100 struct{
    AT7 string
    MS1 string
    MS2 string
    M7 string
}
```

The TCSSMessage in EDI214 specifies the transportation carrier shipment status and consists of a header, a footer and the record of the shipment status details (TCCSSMDetail). TCCSSMDetail has a specific structure and includes four loops labeled Loop1000, Loop1100, Loop1200 and Loop1300. Loop1000 includes information such as business instructions and lading handling requirements. Loop1100 specifies the transportation carrier shipment status. Loop1200 covers the party's basic information (e.g., identification, address). Loop1300 is reserved for the order information details.

In Loop1100 of EDI214, AT7 is a segment reserved for status details, MS1 is for shipment location and MS2 identifies the owner. The timestamp is indicated in the AT7 segment following the X6 EDI code which refers to a shipment that is en route to the delivery location. For example, AT7*X6*NS***20170320*1125 refers to a shipment with normal status (NS) at the given time. The codes MS104 (longitude) and MS105 (latitude) are used to capture the geolocation of the shipment within the MS1 segment.

## 4. Blockchain Model

In the proposed physical distribution framework, the blockchain technology is used to enhance the validity and security of the information being exchanged by the trading partners. Blockchain was first used in the digital currency Bitcoin [14]. In HP3D, it is used to document custody events and the pseudo-real time physical location of the shipment during the transport phase.

Two kinds of blockchain ledgers are implemented: a semi-public ledger and a private ledger. The semi-public ledger consists of monitoring events that are posted by external monitors in order to ascertain the physical location of a given truck. The private ledger consists of shipment information and custody events. Both ledgers are distributed and stored in MongoDB or Couchbase lite depending on the peer type. These two ledgers are discussed next.

### 4.1. Private ledger

Private ledger events are organized in documents consisting of the shipment fields (i.e., Order Info) and four additional derived fields (Figure 3). These derived fields are the hash value of the document, the hash value of the previous document, the hash value of the current block and the hash value of the previous block. Each set of consecutive documents in the sub-ledger are collected into a sub-ledger block.

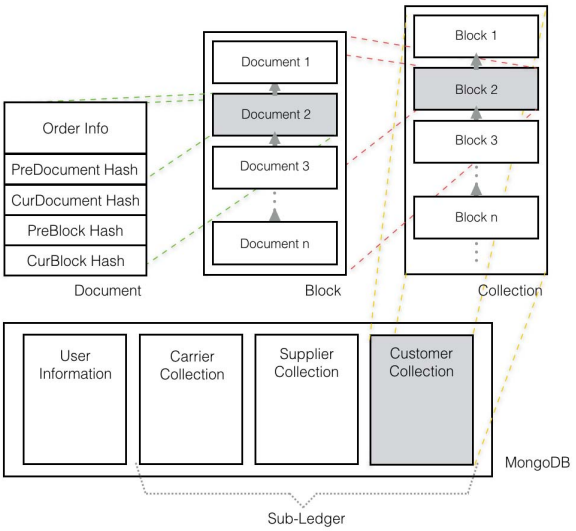


Figure 3. Private Sub-Ledger

The chaining introduced by the previous document hash and the previous block hash are used to

- Enable the traversal of the chain in order to locate the previous event.
- Ensure that the information in the ledger is immutable since changes to any event will render all subsequent events invalid.

The placement of an order or the issue of the delivery note is considered as the genesis event in the sub-ledger. The related information represented in the EDI 214 standard is translated into a byte code and hashed using SHA-256. For the first genesis document, this value is used to populate the current document and the current block hash fields. The remaining hash fields, namely, previous document and previous block hash fields are populated using randomly generated hash values. The next document in the block will also have a document hash value that is generated based on the event information. However, in this case the previous document hash value will be populated with the hash value of the preceding document. The current block hash value is derived from the hash value of the current and previous document hash values. Finally, since this document is still in the first block, the previous block hash value is randomly generated. The sub-ledger is distributed and each of the peers participating in the shipment-centric subnetwork will maintain their copies of the private ledger. The generation of a new block in the private ledger depends on the occurrence of special events (e.g., shipment loaded on truck, GPS location validated, shipment confirmed by customer). The hash values of these special events are used to populate the previous block hash value of the subsequent block.

### 4.2. Semi-Public ledger

The semi-public ledger is maintained by external monitors. A document in the semi-public ledger consists of the Truck ID, a timestamp and the GPS coordinates of the truck (Figure 4). When a delivery truck is in the physical proximity of an external monitor (e.g., another vehicle), a radio communication channel is established between the truck and the monitor. The monitor will send a challenge (a nonce) to the truck, who then signs it with her private key and sends the signature back to the monitor. The external party monitor verifies the signature using the truck's public key. She will then sign the current geolocation and timestamp information of the truck using her private key and post it to the semi-public ledger. Documents and blocks are hashed and chained using the same approach as that of the private ledger. However, in this case the blocks are formed after a fixed number of events.

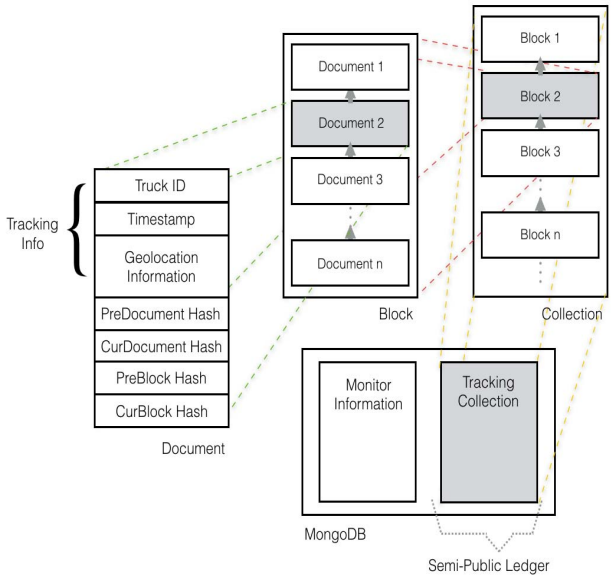


Figure 4. Semi-Public Ledger

The external monitors are only allowed to post to the semi-public ledger. Reading of the ledger is only accessible to the trading partners. The customer, the supplier or the carrier can traverse the semi-public ledger and retrieve the geolocation of a given truck. This information is then used to update the status of the shipment in the private sub-ledger. The public ledger is still in an experimental design

phase. Some of the key issues that need to be addressed is the signing of the information by the external monitors and the distribution of public and private keys. Furthermore, multiple monitors may post to the ledger at very close time intervals. These posting should be consolidated and the related information should have elevated validity based on the number of independent sources.

## 5. Related Work

E2Open [15] and SAP [16] are examples of leading edge supply chain systems that provide well-developed functionalities including pseudo real-time shipment information and interoperability with multiple enterprise resource planning (ERP) systems. These systems are cloud-based and adopt a centralized and often complex software and hardware architectures. Given the large amount of traffic and the variety of stakeholders (i.e., small, medium and large businesses) that need to be supported by a global supply chain system, a centralized architecture has limitations with respect to both scalability and affordability. As a result of these limitations, small and medium business have a difficulty penetrating new and well established markets. The proposed HP3D is based on a P2P architecture that attempts to address these limitations through a decentralized architecture that relies on mainstream technologies which are becoming equally accessible to small, medium as well as large businesses.

One of these technologies is the sensor technology which has evolved rapidly in the recent years making smart devices ubiquitous and affordable. These sensors deliver information about the environment in real-time [17]. For the purpose of this application, sensors are expected to be available throughout the route of the shipment starting from the supplier's warehouse, to the carrier's truck, and ending with the customer's warehouse. The increased availability of sensor technology is also used to enable external monitors to indirectly validate the physical location of a shipment.

The second technology that is being leveraged by the proposed HP3D is that of P2P networks. While P2P architectures have been widely used in several social media applications (e.g., Napster [18], Skype [19] and Spotify [20]), they have not been used in supply chain management. Several configurations of the P2P architecture were proposed in the literature. Gnutella [18], a file sharing application, was the first pure P2P network. In this application, each peer stores its own index of other peers. This configuration is resilient but also prone to increased network traffic [21].

The hierarchical P2P configuration is used by KaZaA [22]. It takes advantage of the locality behavior of data exchanges and relies on a subset of the peers as super-nodes in order to manage indices for a given group of peers [23]. This configuration addresses the increased traffic issue of the pure P2P configuration but at the expense of an increase in network management complexity. Because of the dynamic nature of the proposed HP3D sub-network, this configuration was not suitable as super-nodes may enroll and leave the network throughout the life-cycle of the shipment. When a super-node leaves the network, it has to transfer its role to another node in the network which may create substantial traffic. The hybrid P2P configuration [24] has been adopted by Napster [21] and Spotify [20]. It is also the configuration of choice for the proposed HP3D and includes a central index server that maintains information related to the active nodes in the network.

The third technology used in HP3D is blockchain. Few IT companies (e.g., IBM) started offering blockchain-based solutions for supply chain. However, these solutions tend to be centralized and cloud-based [11], thus avoiding the challenges associated with an open and distributed P2P architecture but also dismissing the intended resilience, scalability and affordability of the P2P network. In general, current research around blockchain focuses on facilitating three main functionalities: tracking (e.g., provenance, proof of origin), transfer (e.g., smart contracts) and payment (e.g., digital currency). Recently, Ethereum [25], a P2P blockchain programming language is being proposed to support all three. HP3D introduces an enhanced blockchain model that combines the use of private sub-ledgers with a semi-public global ledger. This combination allows the cost-effective delivery of ground truth information in a timely manner to all stakeholders.

## 6. Conclusion

This paper introduces a framework that supports supply chain visibility by using a hybrid P2P architecture. Two types of peers are considered, namely X86 peers and mobile peers. Both have the same software architecture but differ in their implementation. Mobile peers have additional functionalities in order to allow them to act as field sensory agents. In addition, an administrative node is introduced in the framework in order to help with mobile peers availability, create a persistent record of the exchanges, and facilitate transfer of information between the organizations ERP systems and the proposed HP3D.

The proposed framework also introduces a blockchain data model which is based on both private and semi-public ledgers. This model takes advantage of the data validity afforded by the public blockchain ledger while protecting the privacy of the trading partners through the shipment-centric private sub-ledger. Future work includes further development and testing of the semi-public ledger and the associated mechanisms for key distribution. Additionally, we would like to investigate suitable sizes for blocks and the time complexity of queries to the semi-public ledger as the number of trucks and the number of monitors increases.

## References


[1] J. T. Mentzer, W. DeWitt, J. S. Keebler, S. Min, N. W. Nix, C. D. Smith, and Z. G. Zacharia, "Defining supply chain management," *Journal of Business logistics*, vol. 22, no. 2, pp. 1–25, 2001.

[2] M. Gifkins, *The EDI handbook: trading in the 1990s*. Online Publications, 1988.

[3] J. Berge, *The EDIFACT standards*. Blackwell Publishers, Inc., 1994.

[4] E. L. Plambeck, "Reducing greenhouse gas emissions through operations and supply chain management," *Energy Economics*, vol. 34, pp. S64–S74, 2012.

[5] J. Holmström, "Business process innovation in the supply chain–a case study of implementing vendor managed inventory," *European journal of purchasing & Supply Management*, vol. 4, no. 2-3, pp. 127–131, 1998.

[6] B. Heaney, "The supply chain visibility: A critical strategy to optimize cost and service," *Aberdeen Group*, vol. 20, 2013.

[7] "Supply chain visibility: Envisioning the broader need," White Paper, Kinaxis, 2014.

[8] J. Meyerson, "The go programming language," *IEEE Software*, vol. 31, no. 5, pp. 104–104, 2014.

[9] E. Plugge, P. Membrey, and T. Hawkins, "The definitive guide to mongodb: the nosql database for cloud and desktop computing," *The expert's voice in open source*, 2010.

[10] D. Delen, B. C. Hardgrave, and R. Sharda, "Rfid for better supply-chain management through enhanced information visibility," *Production and Operations Management*, vol. 16, no. 5, pp. 613–624, 2007.

[11] M. Pilkington, "Blockchain technology: principles and applications," *Research Handbook on Digital Transformations*, 2015.

[12] D. Ostrovsky and Y. Rodenski, "Couchbase lite on android," in *Pro Couchbase Server*. Springer, 2014, pp. 283–292.

[13] D. A. Johnson, B. J. Allen, and M. R. Crum, "The state of edi usage in the motor carrier industry," *Journal of Business Logistics*, vol. 13, no. 2, p. 43, 1992.

[14] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[15] R. Karpinski, "E2open at one," *InternetWeek, August*, vol. 1, 2001.

[16] F. Andera and D. Derringer, "systems, applications, products in data processing" sap: Implications for computer information systems," *Journal of Computer Information Systems*, vol. 39, no. 1, pp. 72–75, 1998.

[17] Z. Pang, J. Chen, D. S. Mendoza, Z. Zhang, J. Gao, Q. Chen, and L. Zheng, "Mobile and wide area deployable sensor system for networked services," in *Sensors, IEEE*. IEEE, 2009, pp. 1396–1399.

[18] S. Saroiu, K. P. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of napster and gnutella hosts," *Multimedia systems*, vol. 9, no. 2, pp. 170–184, 2003.

[19] S. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," *CoRR*, vol. abs/cs/0412017, 2004. [Online]. Available: http://arxiv.org/abs/cs/0412017

[20] G. Kreitz and F. Niemela, "Spotify–large scale, low latency, p2p music-on-demand streaming," in *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P),*. IEEE, 2010, pp. 1–10.

[21] D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov, "Peer-to-peer overlays for real-time communication: security issues and solutions," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 4–12, 2009.

[22] G.-X. Yue, R.-F. Li, and Z.-D. Zhou, "P2p network model with multi-layer architecture based on region,," *Ruan Jian Xue Bao(J. Softw.)*, vol. 16, no. 6, pp. 1140–1150, 2005.

[23] Z. Peng, Z. Duan, J.-J. Qi, Y. Cao, and E. Lv, "Hp2p: A hybrid hierarchical p2p network," in *First International Conference on the Digital Society. ICDS'07*. IEEE, 2007, pp. 18–18.

[24] V. Darlagiannis, "21. hybrid peer-to-peer systems," in *Peer-to-Peer Systems and Applications*. Springer, 2005, pp. 353–366.

[25] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.