


Article

A Distributed Ledger for Supply Chain Physical Distribution Visibility [†]

Haoyan Wu ¹, Zhijie Li ¹, Brian King ¹, Zina Ben Miled ^{1,*} , John Wassick ² and Jeffrey Tazelaar ²

¹ Electrical and Computer Engineering Department, Indiana University-Purdue University, 723 W. Michigan St, SL 160, Indianapolis, IN 46202, USA; haoywu@iupui.edu (H.W.); lizhij@iupui.edu (Z.L.); briking@iupui.edu (B.K.)

² The Dow Chemical Company, 715 East Main St., Midland, MI 48674, USA; JMWassick@dow.com (J.W.); JRTazelaar@dow.com (J.T.)

* Correspondence: zmiled@iupui.edu; Tel.: +317-278-3317

[†] This paper is an extended version of our paper published in the 37th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017.

Received: 9 September 2017; Accepted: 30 October 2017; Published: 2 November 2017

Abstract: Supply chains (SC) span many geographies, modes and industries and involve several phases where data flows in both directions from suppliers, manufacturers, distributors, retailers, to customers. This data flow is necessary to support critical business decisions that may impact product cost and market share. Current SC information systems are unable to provide validated, pseudo real-time shipment tracking during the distribution phase. This information is available from a single source, often the carrier, and is shared with other stakeholders on an as-needed basis. This paper introduces an independent, crowd-validated, online shipment tracking framework that complements current enterprise-based SC management solutions. The proposed framework consists of a set of private distributed ledgers and a single blockchain public ledger. Each private ledger allows the private sharing of custody events among the trading partners in a given shipment. Privacy is necessary, for example, when trading high-end products or chemical and pharmaceutical products. The second type of ledger is a blockchain public ledger. It consists of the hash code of each private event in addition to monitoring events. The latter provide an independently validated immutable record of the pseudo real-time geolocation status of the shipment from a large number of sources using commuters-sourcing.

Keywords: blockchain; supply chain; distributed databases; peer-to-peer networks

1. Introduction

Our main objective is to provide suppliers and customers with validated, near real-time visibility during the physical distribution phase of the supply chain (SC). This phase is concerned with the transport of the goods from the supplier to the customer. There are several existing solutions that support shipment tracking during the physical distribution phase of SC. These solutions are often populated with tracking information from a single source, the carrier, and suffer from a restricted visibility to other stakeholders. Indeed, information is shared through updates provided by the carrier as and when deemed necessary. Moreover, the information being shared is not validated by an independent source. These trust and transparency issues may not affect trading among large businesses with fully integrated inbound and outbound logistics networks [1]. However, the model fails when either the customer or supplier are small or medium businesses and have to rely on load sharing and multiple carriers during shipment.

This paper introduces a framework that delivers online shipment tracking information to all stakeholders during the distribution phase of SC. Information flow and exchange is supported by two types of distributed ledgers: private ledgers and public ledgers. Each shipment is associated with a specific private ledger that is only shared by the trading partners involved in the shipment. This design choice seems to go against the principle of transparency being promoted in this paper. However, we deem that it is necessary to retain a level of privacy when the traded products are high value, hazardous or may have dual use (e.g., pharmaceutical and chemical products). The private ledger consists of custody events related to a specific shipment. An example custody event can be the transfer of the shipment from the supplier to the carrier or from the carrier to the customer.

The second type of ledger is the public ledger. There is only one public ledger in the proposed framework and it includes events posted by external monitors and the hash values of all the custody events posted to the private ledgers. The monitors are commuters that can validate the geolocation of the truck, which is then associated, by the trading partners, to their specific shipments through the information in the private ledger. The commuters enhance the validity of the tracking information as well as support pseudo-real time delivery of this information. In addition to monitoring events, the public ledger also includes the hash value of the private custody events. As mentioned above, custody events are only shared by shipment-specific trading partners. However, a public record of these events is maintained by posting their hash values to the public ledger. This public record provides an additional validation mechanism for custody events.

While this paper focuses on the specific application of the blockchain technology to support supply chain visibility in the physical distribution phase, it also attempts to address the broader divide between completely private/permissioned and completely public/open blockchain architectures. The advantages and disadvantages of either type of architecture have been discussed in [2]. However, more research is needed towards trust architectures that combine the two types of ledgers [3], especially when the protection of sensitive information is critical.

The architecture of the proposed system is distributed and uses a hybrid peer-to-peer communication model [4,5]. This paper specifically focuses on the data model and data representation. The underlying data model uses the EDI-214 (Electronic Data Interchange) standard [6] translated into JSON (JavaScript Object Notation) [7] in order to maintain inter-operability with existing SC information systems. Other aspects of the framework, such as those related to user authentication and cryptographic key management, are out of the scope of this paper and as such not discussed in details.

Section 2 of the paper is a review of previous related work. Section 3 introduces the framework and discusses the data representation used for the ledgers and the underlying events as well as the processes that enable the posting of these events to both the private and public ledgers. Section 4 covers the implementation of the framework and exemplifies the framework in operation using a test scenario. Section 5 concludes this paper by highlighting the main contributions and summarizing directions for future work.

2. Related Work

This section presents a perspective on the evolution of SC management systems as well as a review of current state of the art in the blockchain technology and its applications.

Major advances have been achieved in the past few decades in the area of SC management systems. Increased maturity and adoption levels of ERP (Enterprise Resource Planning) solutions allowed companies to effectively manage their internal logistics processes in warehousing, order management, financing, billing, etc. Moreover, increasing demand for national and international trade led to the emergence of B2B (Business to Business) platforms that are powered by supply chain operating networks (SCONs) [1]. These networks are often supplier-centric and serve a specific industry sector. Examples of commercially available SCONs solutions include E2Open [8] and SAP (Systems Applications and Products) [9,10]. SCONs can be integrated with the organization's ERP and enable information exchange among trading partners (Figure 1) using standard EDI [6,11] messaging.

They are very efficient at exposing a target set of transactions (e.g., purchase orders and bill of lading) from a given member to other trading partners. However, SCONs have only recently started to adapt to the shift from supplier-managed physical distribution to third-party-managed physical distribution. Indeed, in the past decade, large manufacturers started outsourcing their physical distribution to third party logistics operators [10]. This allowed manufacturers to focus on their core business. However, along the expansion in both national and international trade, this trend created complex logistics networks. In order to overcome these complexities, Transportation Management Systems (TMS) started to evolve from the traditional fleet management systems for vehicle tracking to include additional services such as shipment-level tracking [12]. This evolution, in our opinion, is still limited for the following reasons: (1) tracking information is not validated, (2) continuity across multiple carriers is not guaranteed and (3) tracking information is still managed by the carrier and disclosed as needed to other stakeholders. These limitations are easy to overcome for tier 1 enterprises with extended resources and large product volumes as they can dictate the rules of engagement. Small and medium enterprises (SMEs) have low product volumes and relatively a large number of partners. As a result, these limitations constitute a major barrier and restrict the access of SMEs to different SCONs [13] due to cost and difficulties in introducing new SC management practices [12,14], which, in turn, prevents them from easily penetrating new markets. The proposed solution is aimed at addressing the above barriers and providing a distributed solution for SC physical distribution visibility by using the emergent blockchain technology [15].

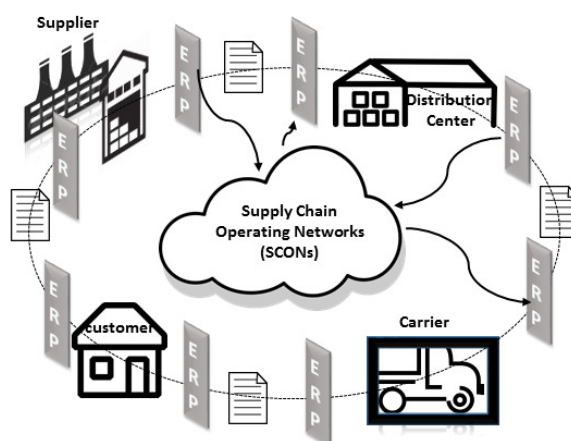


Figure 1. Current supply chain operating networks (SCONs).

As initially proposed in [15], a blockchain is a distributed public database (ledger) that maintains a continuously growing list of transactions which are organized in blocks and secured from tampering. Two types of transactions are supported: genesis transactions that create value and transfer transactions that transfer value from one party to another. Each transaction is digitally signed by the issuer and posted to the ledger. A group of transactions are then collected into a block that is validated by a third party (a miner). Each block in the chain is immutable since it is linked to its predecessor and any change to any of the blocks invalidates all the blocks downstream in the chain. Furthermore, the more mature the block is (i.e., the longer it has been in the ledger chain), the greater is its integrity. Each participant in the network keeps a copy of the ledger and every time a new block is created, it is broadcasted to all the participants that add it to their local copy of the ledger.

The blockchain technology gained the interest of a large number of researchers. A recent search on Scopus with the keyword “blockchain” revealed nearly 600 articles since 2014. This is the starting point of a structured literature review process that has been proposed in [16–18]. A taxonomy for the various blockchain architectures proposed by these researchers as well as the advantages and limitations of each of these architectures are offered in [2]. Primarily, these trust architectures can be

classified along three main dimensions: (1) centralized versus distributed; (2) private/permissioned versus public/open; and (3) on-chain versus off-chain data storage.

Although primarily designed for a distributed architecture, the blockchain technology can be implemented on a centralized architecture. The distributed architecture often adopts a peer-to-peer communication model [19–21]. Centralized, cloud-based architectures avoid the security and privacy challenges associated with a distributed architecture but also dismiss the intended resilience, distributed governance, scalability and affordability of the distributed architecture. IBM (International Business Machines) [22] offers a centralized, permissioned blockchain end-to-end SC management system. Blockfreight [23] is a public distributed blockchain solution for SC container tracking.

A blockchain ledger can be private (permissioned) or public (open). Private ledgers tend to be implemented over a centralized architecture and public ledgers tend to be implemented over a distributed architecture. In addition to the above two examples [22,23], several other private and public blockchain SC solutions have been proposed in the literature. Actually, a refinement of the above search on Scopus with the additional keyword “supply chain” returned approximately 70 articles. For example, the ledger in [24] is public and supports information sharing in SC demand planning from the perspective of SMEs. The ledger proposed in [25] is private. It supports B2B trading and facilitates direct manufacturer-customer transactions and payment services. Typically, SC applications of the blockchain technology include proof of origin and traceability [24,26], inventory and demand levels sharing [25] and enforcement of regulations for pharmaceutical [27] and food products [28].

Privacy is especially of concern when sharing sensitive information. For instance, some companies may not be willing to reveal their production capacity to their competitors [25]. In the health sector, sharing of medical records is regulated by strict privacy rules. An alternative approach to a private ledger architecture consists of the use of off-chain data storage instead of on-chain data storage. For example, in [29], a blockchain ledger was proposed for sharing of patients’ medical records. The medical records are securely stored on the cloud separately from the ledger. Requests for access are then posted to a public ledger and consensus nodes are responsible for fetching pending requests from this public ledger, verifying the validity of these requests and posting them to a private ledger for traceability purposes.

Information exchange by using a completely public or a completely private ledger architecture may not be able to address the requirements of various applications including SC applications. A solution based on a coordinated use of both types of ledgers is needed. Sensitive data can be stored in a private ledger, whereas data that require a high trust level can be stored in a public ledger. The two types of ledgers can be designed to limit the information accessible by each stakeholder without reliance on central governance. The proposed framework advocates this approach and can be used beyond the current SC physical distribution application. Some of the emerging applications that can benefit from this model include finance, government services [30,31], IOT (Internet of Things) [26] and regulation compliance [27].

3. Framework Design

The proposed framework consists of a set of dynamic and private sub-ledgers and a central public ledger. The sub-ledgers are private to the trading partners and a sub-ledger is created for each shipment. Participation in the sub-ledger is based on the partners involved in the execution of the corresponding purchase order. The public ledger, on the other hand, is open to all and contains tracking information for all trucks’ transporting shipments in addition to a record representing the hash value of each private event. The segregation of data record and their hash value in different ledgers was also proposed in [32] in order to protect personal data in a service-centric blockchain application. Both the sub-ledgers and the public ledger are distributed and each participating node keeps a copy of the ledger of interest. This section describes the data model used for both types of ledgers and the mechanisms used to keep the information in these ledgers up-to-date.

3.1. Architecture Overview

The architecture of the proposed hybrid peer-to-peer physical distribution (HP3D) framework [33] consists of a set of sub-networks, each associated with a specific shipment. The participants in the sub-network are the trading partners involved in the shipment. Furthermore, the sub-network is created when the order is placed and terminates when the goods are delivered to the customer.

There is also a global network that is open to all partners as well as third-party monitors. This global network is persistent and does not terminate. It is used as a timestamp and a proof of record for the events related to all shipments.

There are four types of participants in HP3D: index server, peers, administrative nodes and external monitors (Figure 2):

- The index server is a central directory that maintains the address of all the nodes in the network. It also assigns a unique ID to each participant.
- Peers are nodes that can take on different roles in different shipments (e.g., customer, supplier and carrier).
- The administrative node is a special peer node. Each trading partner has one administrative node, which is responsible for communicating with the ERP internal to the organization. This node participates in all sub-networks involving the associated partner and maintains a permanent record of all information exchanges related to the partner's shipments.
- Third party external monitors are responsible for the validation of the geolocation of the shipments and the posting of this information to the public ledger.

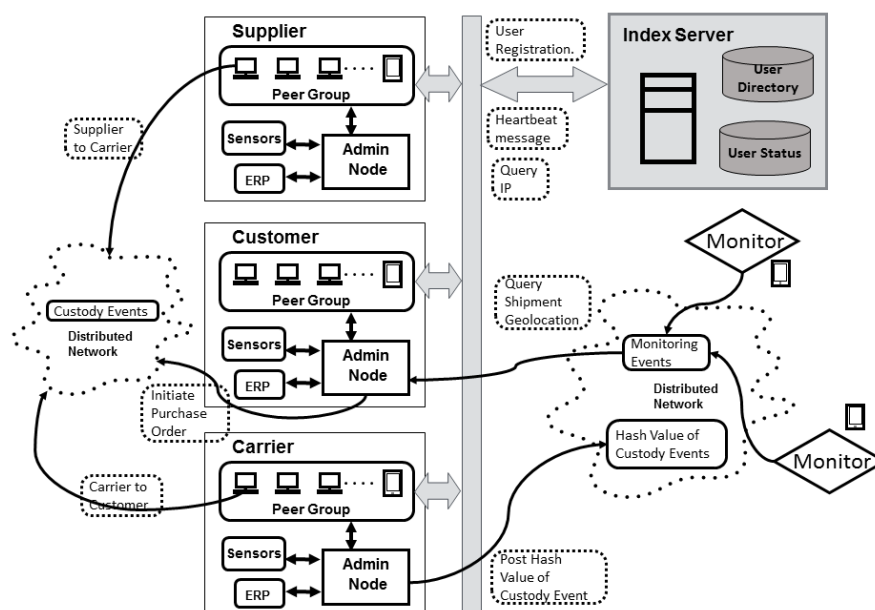


Figure 2. Interactions among the participants in the proposed framework.

All the nodes post information to either the private or the public ledgers in the form of events. The different types of events supported by the proposed framework are discussed next.

3.2. Events

The proposed framework supports three types of events to document the status of a shipment:

- The genesis event indicates the start of a shipment (i.e., the issue of a purchase order). This event is similar to the genesis event in Bitcoin [15]. It is initiated by the administrative node of the customer and broadcasted to all trading partners. Each partner that is participating in the shipment stores

this information into its local database in the form of a document. The details of the genesis event are only accessible to the trading partners. However, a hash value calculated from the genesis event is also posted to the public ledger by the event generator, in this case, the customer.

- The custody event is a record of the custody status of the shipment. The custody can remain with the current holder of the shipment or show a transfer from one participant to another (e.g., shipment transferred from supplier to carrier, shipment delivered to customer by carrier). In addition to the genesis event, the custody events form the shipment-centric private ledger that is shared among the supplier, carrier and customer for a given shipment. Similar to the genesis event, the hash value of each custody event is also calculated and posted to the public ledger by the event generator.
- The monitoring event indicates the geographical location of a shipment. This information is generated by external monitors when trucks are physically near the monitors and an information exchange is executed between the monitors and the trucks in order to document this physical proximity. The monitoring events are posted to the public ledger by the external monitors.

In the case of the public ledger, a group of events are chained to form a block, and, in turn, blocks are chained to form the ledger. The posting of events to both private and public ledgers is depicted in Figure 3.

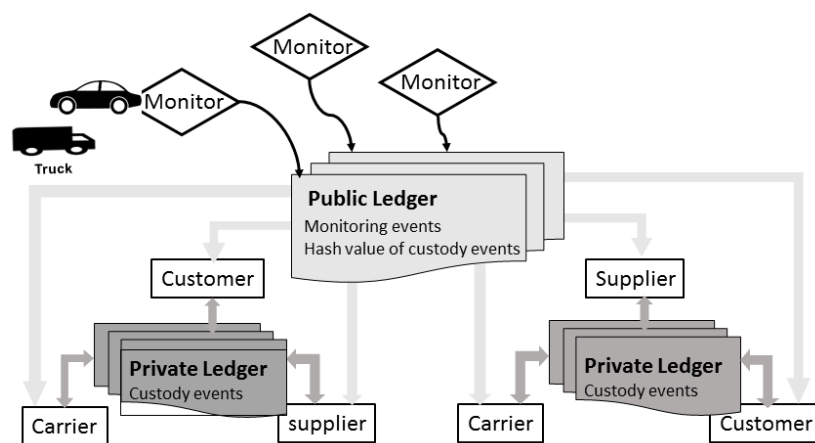


Figure 3. Posting of events to the private and public ledgers.

3.3. Data Structures

HP3D has three kinds of data structure for exchanging and storing events and blocks. These are *PrivateGCEvents*, *PublicEvents* and *Blocks*.

PrivateGCEvents is the data structure used for the genesis and custody events that are exchanged across trading partners. This data structure (Figure 4) has three fields, a unique ID, a timestamp for the event and a sub-structure containing the details of either a genesis event or custody event. This sub-structure is based on the EDI-214 standard. The EDI-214 standard message is traditionally represented as a string and a parser was used to translate this message from a string representation to a JSON representation and vice versa.

```
type PrivateGCEvents struct{
    EventID bson.ObjectId 'bson:"_id,omitempty"'
    Timestamp string
    EDI214 InterchangeEnvelope
}
```

Figure 4. Private genesis and custody events data structure.

PublicEvents (Figure 5) is the data structure used in the public ledger. In the case of genesis and custody events, *EventHash* is the field that records the hash value of the event. This hash value is derived from the source event in the EDI-214 format. The trading partners can check the validity of these two types of events by comparing the hash value in the public ledger with the one computed from the *PrivateGCEvents* data structure in their private ledger.

```

type PublicEvents struct{
    EventID bson.ObjectId 'bson: "_id,omitempty"'
    Timestamp string
    MonitorData
    EventHash [32]byte
    PreEveHash [32]byte
    CurEveHash [32]byte
}
type MonitorData struct{
    MonitorID bson.ObjectId 'bson: "_id,omitempty"'
    TruckID bson.ObjectId 'bson: "_id,omitempty"'
    GeoInfo GPS
}

```

Figure 5. Public events data structure.

Monitoring events use the *MonitorData* and *EventHash* data structures. *MonitorData* (Figure 5) is a sub-structure that contains the information of the monitor, the truck and its geolocation. *EventHash* is computed from the *MonitorData*. The fields *PreEveHash* and *CurEveHash* are hash values for chaining the events and building the blocks in the public ledger. These two values are calculated locally by each node.

Events in the public ledger are grouped into blocks using a data structure called *Blocks* (Figure 6), where *BlockID* is a unique identifier for the block. Since a block may contain multiple chained events, the *Event* field is an array of *PublicEvents*. The field *Timestamp* corresponds to the time the block is created. The *Nonce* is an arbitrary integer value computed for each block. It is used to increase the complexity of computing the hash value. *PreBloHash* has the hash value of *CurBloHash* from the previous block.

```

type Blocks struct{
    BlockID bson.ObjectId 'bson: "_id,omitempty"'
    Event []PublicEvents
    Timestamp string
    Nonce int
    PreBloHash [32]byte
    CurBloHash [32]byte
}

```

Figure 6. Blocks data structure.

3.4. Blockchain

A blockchain ledger can be considered as a distributed database that holds a continuously increasing list of events grouped into blocks [34]. As previously mentioned, the proposed framework has two types of ledgers: a private ledger and a public ledger (Figure 3). The private ledger contains the details of the genesis events and the custody events. These events are not chained thereby limiting the computing requirements for nodes belonging to the trading partners. Moreover, the validity of each of these events can be easily verified by comparing the hash value of the source event in the private ledger to the corresponding hash value in the public ledger. Therefore, in the proposed design,

the public ledger acts as a reference and proof of validity for the genesis and custody events in addition to being a permanent record for monitoring events.

The local database of each node in the network may include up to four collections: *PrivateEvents*, *tempPublicEvent*, *Blocks* and *tempBlocks*:

- *PrivateEvents* captures the details of genesis and custody events.
- *tempPublicEvent* is used for temporary storage of public events that are not yet part of a block in the public ledger.
- *Blocks* is used for chained blocks. The longest chain is accepted by every node in the network as the public ledger of record.
- *tempBlocks* is used when multiple blocks are received at the same time, or when the *PreBloHash* of the received block does not match the latest block in the local database of the node.

The local database of each monitor has two collections: *tempPublicEvent* and *Blocks*. These collections are similar to the ones used for the trading partners nodes discussed above. Monitors are responsible for maintaining the public ledger. Therefore, they do not generate genesis events or custody events and do not need to maintain the *PrivateEvents* collection.

3.5. Validation

There are two kinds of information validation in the proposed system. The first is the validation of events. The second is the validation of blocks. The events are checked every time they are received. Therefore, there is no need to check them again when they are chained or combined into a block in the public ledger.

The genesis event indicates a new shipment. The administrative node of the customer is the only node that can generate a genesis event. Since this event initializes a new shipment and establishes the order ID, the customer, the supplier and the carrier, the genesis event does not need to be validated.

The custody event involves two participants from two different companies. This event is signed with the participants' private keys before it is shared. Normally, node A of the first participant signs the event first, then node B of the second participant signs the message that has been previously signed by node A. Subsequently, node B sends the doubly signed custody event and is considered as the event generator. The details of a custody event are broadcasted to all trading partners in a given shipment by the event generator. The nodes who receive the private custody event check the message by using node A and node B's public keys. This double-signature mechanism improves the security of the private ledger. If the event is accepted, the recipient node will save the custody event into its local database. The node also calculates and stores the public version of the event consisting of the hash value of the private event in its local copy of the public ledger. The public version and the private version of the custody event share the same *EventID*. The event generator is also responsible for sharing the public version of the event with non-trading partners.

A monitoring event is generated by an external monitor. External monitors do not have any information about the shipment except for the *truckID* and geolocation. The information in the monitoring event is validated through crowd-sourcing since multiple monitors are expected to report the geolocation of a single truck.

The second type of validation is with respect to blocks. In the proposed system, any node can propose a candidate block. In order to limit the number of proposed blocks that a node can put forward, the node needs to find a nonce that satisfies a hash value with a given format. For example, the hash value may include a number of leading zeros making the process of computing a hash value for a block computationally expensive. After the proposed block is received, other nodes validate the events in the block. They accept the block only if the *nonce* is satisfied and all the public events have values in the *EventHash* field that are identical to the ones stored in their local databases.

3.6. Processes

Three main processes are used to manage events and blocks:

- The *Private Event Process* supports the sharing of private events among trading partners in a shipment.
- The *Public Event Process* allows the exchange and the posting of public events to the public ledger.
- The *Build Block Process* combines multiple events into a block and validates each block in the public ledger.

The workflow for the private event process is shown in Figure 7. Two cases are handled by this process: genesis events and custody events. In the first case, the administrative node of the customer generates a genesis event by extracting data from the company's ERP. For example, node A and node B in Figure 7 represent an administrative node and a regular node, respectively. Once the message is received from the ERP, node A generates a unique ID for the event. This administrative node will then broadcast the event to the other partners.

In the second case, a regular node triggers a custody event when the underlying message is signed by two participants. For example, when a shipment is being transferred from a supplier to a carrier, an employee of the supplier signs the custody event with his private key to confirm the event. Subsequently, an employee of the carrier signs the custody event and sends it to the trading partners.

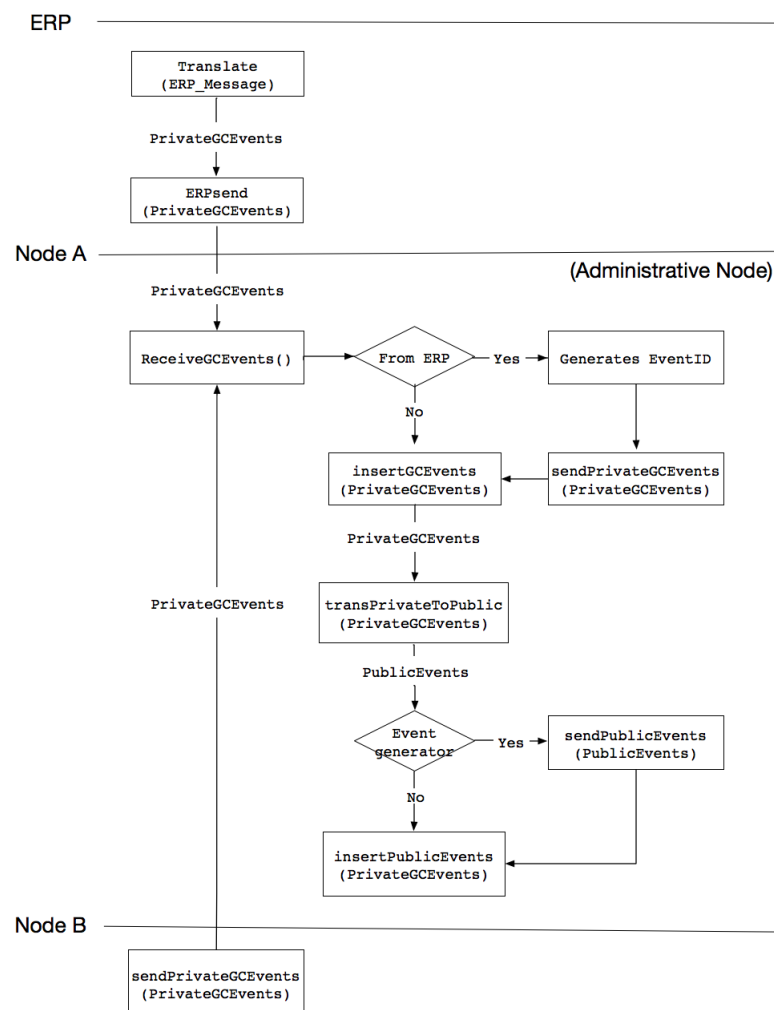


Figure 7. Private event process.

The next step in the process in both cases consists of generating the public event associated with this private event. The hash value of the private event is computed by the event generator and the resulting public event is sent to non-trading nodes. The process concludes by having the event generator store the public event in its local copy of the public ledger. Private events are only shared among trading partners. Therefore, a monitor will never receive a private event. It will only receive the hash value of the private event when it is posted to the public ledger by the private event generator.

3.6.1. Public Event Process

A public event can be one of two types (Figure 8). The first type is the hash value of a private genesis or custody event that has been generated through the process described above. The second type is a monitoring event. Any participant in the network is able to send or receive a public event. The public events received are chained together to form the public ledger.

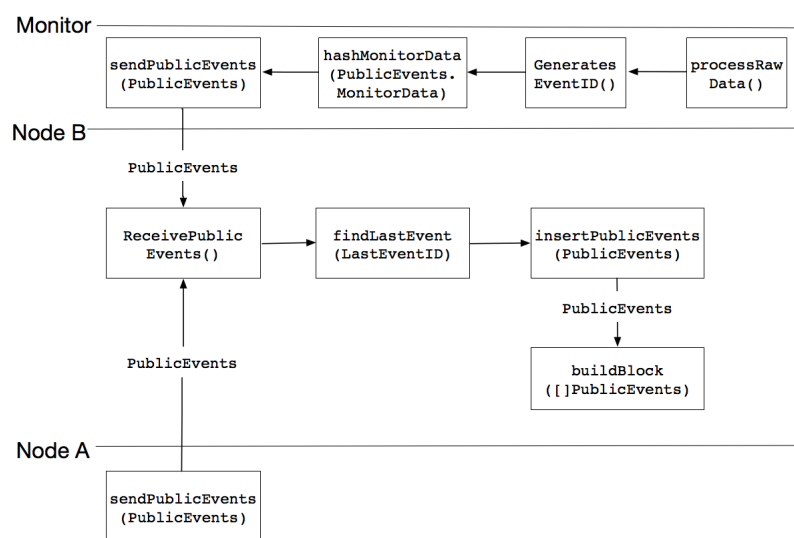


Figure 8. Public event process.

In the case of a monitoring event, data is created when the monitor and the truck exchange a message indicating physical proximity. The monitor generates a unique ID for the event. It will also calculate the hash of the event and store it in the *EventHash* field of the *MonitorData* data structure. Finally, the monitor broadcasts the event to the participants of the public ledger.

Once a public event is received, it is inserted in the *tempPublicEvent* collection and chained to the previous event in the collection (i.e., the *CurEveHash* of the previous event is used as the *PreEveHash* of the new event.) Once the number of events in the *tempPublicEvent* collection exceeds a preset value, a new block can be proposed.

Since each node may receive public events in different order, the *PreEveHash* of the events may differ, which results in different values of the *CurEveHash*. However, the *EventHash* is independent of the order in which the events are received. Furthermore, it is consistent since it is computed by the event generator. The *Building Block Process* that is described next is responsible for receiving the block and checking the validity of each *EventHash*.

3.6.2. Build Block Process

Any node or monitor can propose a block (Figure 9). A block is built by using the events in the order they are received by the node. If a nonce is found, the node broadcasts its proposed block to other nodes. When the other nodes receive the candidate block, they can check the validity of the block by recomputing the *CurBloHash* using the nonce provided. The nodes then check each event's *EventHash* value in the block by comparing it to the local database. If the result is different, the proposed block

will be discarded. If the candidate block is confirmed, the *CurBloHash* field of the previous block is compared to the current block's *PreBloHash*. The block is stored and chained in the local database, if these two fields match. Otherwise, the candidate block will be saved in a temporary collection called *tempBlocks*. The recipient node may be missing a block or there could be multiple branches of the public ledger. Resolving these ledger collision and fork issues is the subject of future work.

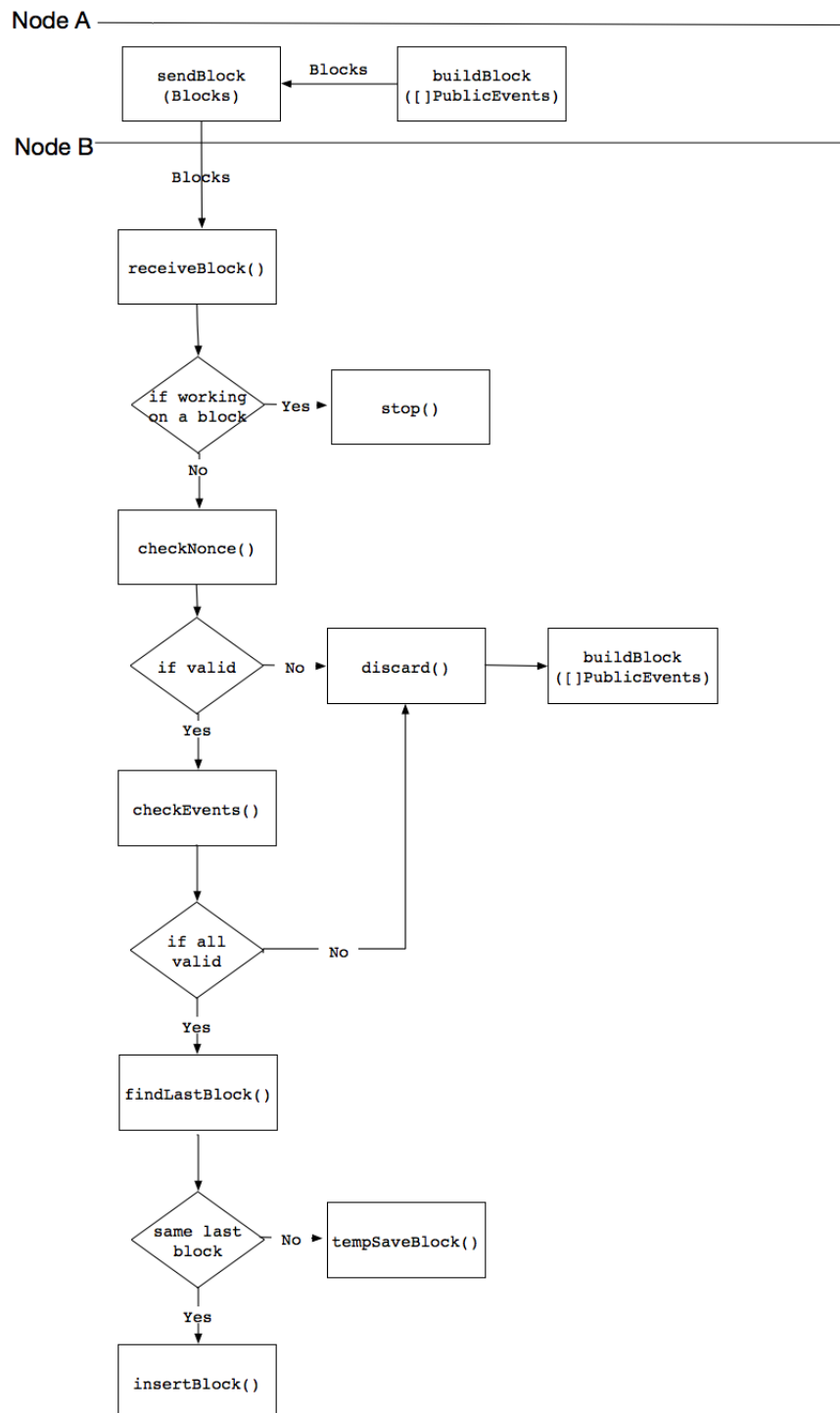


Figure 9. Build block process.

4. Implementation and Discussion

There are three kinds of nodes in the proposed system: X86 nodes running on X86 platforms, mobile nodes running on Android platforms and administrative nodes that are also running on X86 platforms but have extended functionalities. These nodes may participate in the private ledger, the public ledger or both.

The X86 nodes and administrative nodes have the same architecture consisting of three tiers: presentation tier, middle tier and data tier. The monitors only have a middle tier and a data tier. The middle tier of an X86 node application is implemented using Golang [35] and Mgo (MongoDB [36] driver for Golang). The monitors can receive signals from sensors, generate and broadcast public events and update their local database. The X86 nodes and administrative nodes also have these functionalities in addition to the ability to handle incoming user requests from the presentation tier and to generate and send private events to their partners. All nodes can build blocks. The ledgers are stored in the data tier and managed by the middle tier.

In order to demonstrate the functionalities of the public ledger and the private ledgers, a testing environment consisting of three nodes and one monitor was developed. The testing focuses on the exchange between the nodes. For clarity, exchanges between the nodes and the index server have been omitted. Node A in the test scenario is an administrative node that belongs to company A. Nodes B and C belong to company B and C, respectively. The following five events are considered in the test scenario:

- Event 1: A genesis event that is generated by node A. It is an event related to order 1 in which company A and company B are participating.
- Event 2: A custody event that is sent by node B. This is an event related to order 2, which is being shared with company B and company C.
- Event 3: A monitoring event that is generated by an external monitor. It consists of a public event that is sent to all nodes in the network. It represents a geolocation update for order 2.
- Event 4: A custody event that is generated by node B. This is an event related to order 1 in which company A and company B are participating.
- Event 5: A monitoring event that is generated by an external monitor. It consists of a public event that is sent to all nodes in the network and represents a geolocation update for order 1.

Two sub-networks are created by the above test scenario. One is shared by node A and node B for order 1. The other sub-network is shared by node B and node C for order 2. These subnetworks are private to the trading partners. A global network is also needed to support the exchange of public events that form the public ledger.

A simulator was created to trigger the five events mentioned above. For a genesis event, the simulator mimics an ERP that is responsible for sending the genesis event information to an administrative node. For a custody event, the simulator verifies and signs the event. For a monitoring event, the simulator acts as a sensor that sends geolocation data to the external monitors. In addition, for testing purposes, the block size was set to 4. Therefore, when there are four public events in the *tempPublicEvent* collection, these events are used to form a candidate block.

The simulator is used to trigger event 1, which is received by node A. This latter node assigns a unique ID to this genesis event. Furthermore, since this is a private event, node A will only send this event to node B. It will then insert the event into its local database in the *PrivateEvents* collection. In this case, node A is considered as the event generator. Thus, it will also compute the *EventHash* and post it to the public ledger. Node B receives the detailed genesis event from node A and saves it into its local database. Node C and the monitor receive the corresponding public event and insert it into their *tempPublicEvent* collection after chaining and calculating *CurEveHash*. Since this is the first event in the database, the *PreEveHash* field has the same hash value as *EventHash*. Figures 10 and 11 show the *PrivateEvents* collection and the *tempPublicEvent* collection in node A after event 1, respectively.

There is one document in the *PrivateEvents* collection. This private event is shared by node A and B. Therefore, node B also has the same document in its *PrivateEvents* collection. For all nodes and the monitors, the translated event is stored in the *tempPublicEvent*. Given that this is also the first event in the public ledger, all *tempPublicEvent* collections across all nodes (Figure 11) are the same.

```
> db.PrivateEvents.find()
{ "_id" : ObjectId('595514e51d41c826a2b0894b'),
  "timestamp" : "2017-1-1 10:00PM",
  "eventdetails" :
    { "status" : "Order Placed",
      "gpscordsx" : "", "gpscordsy" : ""
    }
}
```

Figure 10. PrivateEvents collection of node A after event 1.

```
> db.tempPublicEvent.find()
{ "_id" : ObjectId('595514e51d41c826a2b0894b'),
  "timestamp" : "2017-1-1 10:00PM",
  "monitordata" :
    { "monitorid" : "",
      "truckid" : "",
      "geoinfo" :
        { "gpscordsx" : "",
          "gpscordsy" : "" }
    },
  "eventhash" : BinData(0,"/IRbc+0y6HvQPHAKwQPIxxG1FAXkVe2WB1
    OXWQzSe/I="),
  "preevehash" : BinData(0,"/IRbc+0y6HvQPHAKwQPIxxG1FAXkVe2WB
    10XWQzSe/I="),
  "curevehash" : BinData(0,"A6NwbhE4Gix7quTB+g1uCrNu/agoTDJbJ
    kk0QZf7EVO=") }
```

Figure 11. tempPublicEvent collection of node A after event 1.

The simulator is then used to trigger event 2. Subsequently, node B receives the custody event. It assigns a unique ID to the event and executes the same steps previously executed by node A for event 1. Node A does not participate in event 2. Therefore, it does not receive the private event and the private ledger of node A remains unchanged (Figure 10). However, node A receives the corresponding public event as any other node in the network. This is shown in Figure 12.

In step 3, the simulator is used to trigger the third event, which is a monitoring event. Monitoring events are public events. The monitor assigns a unique ID to the public event and computes the *EventHash*. It saves the event into its local *tempPublicEvent* collection and sends it to nodes A, B and C. The value of *CurEveHash* for the previous event in the *tempPublicEvent* is used as the *PreEveHash* for the current event. The *tempPublicEvent* collection of node A after the receipt of public event 3 is shown in Figure 13.

```

> db.tempPublicEvent.find()
{ "_id" : ObjectId("595514e51d41c826a2b0894b"),
  ...
  "eventhash" : BinData(0,"/IRbc+Oy6HvQPHAKwQPIxxGlFAXkVe2WB
    10XWQzSe/I="),
  "preevehash" : BinData(0,"/IRbc+Oy6HvQPHAKwQPIxxGlFAXkVe2W
    B10XWQzSe/I="),
  "curevehash" : BinData(0,"A6NwbhE4Gix7quTB+giuCrNu/agoTDJb
    Jkk0QZf7EVO=")}
{ "_id" : ObjectId("595515221d41c811406c398f"),
  ...
  "eventhash" : BinData(0,"5mDGKxSaXZCLmkfugYxpqbhoaIdzUJAna
    QnTjgsjwgo="),
  "preevehash" : BinData(0,"A6NwbhE4Gix7quTB+giuCrNu/agoTDJb
    Jkk0QZf7EVO="),
  "curevehash" : BinData(0,"tNELVhCUwHQzn0cyQ1spp3wxHtIcNqOh
    L/iSKi4vDCO=")}

```

Figure 12. tempPublicEvent collection of node A after event 2.

```

{ "_id" : ObjectId("595515491d41c8492a5ea800"),
  "timestamp" : "2017-3-4 07:30PM",
  "monitordata" :
  { "monitorid" : "485736251d4ac4594a5ba631",
    "truckid" : "948265738a61e8293a6aa766",
    "geoinfo" :
    { "gpscordsx" : "16",
      "gpscordsy" : "63"}},
  "eventhash" : BinData(0,"oNAFBBMrmUP36CYh5UdIqz5IAFQsNG+Lwp
    mRRB08Kcw="),
  "preevehash" : BinData(0,"tNELVhCUwHQzn0cyQ1spp3wxHtIcNqOhL
    /iSKi4vDCO="),
  "curevehash" : BinData(0,"iQ9VPUYwZS/M26aUbZEicbs01pzoNIhke
    EZQwIuATSU=")}

```

Figure 13. New document in tempPublicEvent collection of node A after event 3.

Event 4 is the same as event 1, except that the event is sent by node B. Broadcasting event 5 is exactly the same as for event 3. When event 5 is received, all nodes including the monitors start building the new block. They assign a unique ID, calculate the nonce for the new block and chain it to the previous block. The resulting block is considered as a candidate block until it is validated by other nodes. This block is shown in Figure 14.


```

> db.Blocks.find()
{ "_id" : ObjectId("595516071d41c826a2b0894c"),
  "event" : [
    { "_id" : ObjectId("595514e51d41c826a2b0894b"),
      ...
      "eventhash" : BinData(0,"/IRbc+Oy6HvQPHAKwQPIxxGlFAXkVe2W
        B10XWQzSe/I="),
      "preevehash" : BinData(0,"/IRbc+Oy6HvQPHAKwQPIxxGlFAXkVe2
        WB10XWQzSe/I="),
      "curevehash" : BinData(0,"A6NwbhE4Gix7quTB+g1uCrNu/agoTDJ
        bJkkOQZf7EVO=")},
    { "_id" : ObjectId("595515221d41c811406c398f"),
      ...
      "eventhash" : BinData(0,"5mDGKxSaXZCLmkfugYxpqbhoaIdzUJAn
        aQnTjgsjwgo="),
      "preevehash" : BinData(0,"A6NwbhE4Gix7quTB+g1uCrNu/agoTDJ
        bJkkOQZf7EVO="),
      "curevehash" : BinData(0,"tNELVhCUWHQzn0cyQ1spp3wxHtIcNq0
        hL/iSKi4vDC0=")},
    { "_id" : ObjectId("595515491d41c8492a5ea800"),
      "timestamp" : "2017-3-4 07:30PM",
      "monitordata" :
      { "monitorid" : "485736251d4ac4594a5ba631",
        "truckid" : "948265738a61e8293a6aa766",
        "geoinfo" :
        { "gpscordsx" : "16",
          "gpscordsy" : "63"}},
      "eventhash" : BinData(0,"oNAFBBMrmUP36CYh5UdIqz5IAFQsNG+L
        wpmRRB08Kcw="),
      "preevehash" : BinData(0,"tNELVhCUWHQzn0cyQ1spp3wxHtIcNq0
        hL/iSKi4vDC0="),
      "curevehash" : BinData(0,"iQ9VPUYwZS/M26aUbZEicbs01pzoNIh
        keEZQwIuATSU=")},
    { "_id" : ObjectId("5955156c1d41c811406c3990"),
      ...
      "eventhash" : BinData(0,"WYNtQT18tZJmIk14D7n9EzH80b2q/KFa
        ylKNyXpGcSM="),
      "preevehash" : BinData(0,"iQ9VPUYwZS/M26aUbZEicbs01pzoNIh
        keEZQwIuATSU="),
      "curevehash" : BinData(0,"s1sji/sBHv5JNN8IKoH9ZaWh4dFZH16
        9sQKS44JVbXs=")}],
    "timestamp" : "2017-06-29 15:00:23.512742763 +0000 UTC",
    "nonce" : 8,
    "preblohash" : BinData(0,"l+hAiPl/vyaY31rqxYijAZOGxJLdOKyy+lgZ
      OF/adrE="),
    "curblohash" : BinData(0,"HS07KblHONS73Y4clZlqBFE1AhxAxbDOexPA
      rcUG3SY=")}

```

Figure 14. Candidate block in node A.

5. Conclusions

The proposed framework leverages distributed databases, the blockchain technology and the hybrid peer-to-peer communication model in order to deliver independently validated shipment tracking information to all stakeholders in pseudo real-time. Through a combined private-public ledger architecture, the solution takes into account the privacy requirements of trading partners while providing the necessary SC visibility for critical decision-making. Carefully managing the trade-off

between privacy and transparency in blockchain applications is essential to the widespread adoption of this technology. Indeed, a solution based entirely on a public ledger may not be attractive to various industries and public sectors. Similarly, a permissioned/private ledger limits the intended utility of the technology. The approach proposed in this paper demonstrates the potential of combining both types of ledgers to realistically address a gap in supply chain. Interestingly, while it was not an intended target, the proposed model may be able to also document the carbon footprint of a given organization during the physical distribution phase. We also believe that the proposed model can be extended to other application areas particularly with respect to regulatory compliance in the food and pharmaceutical industries.

The proposed framework was only tested on a limited scale in a laboratory environment. Several challenges may emerge when it is deployed in a network with a large number of nodes. Some of these challenges include resolving block collisions and forks. Limited research is available today in this area [16] and additional studies are needed to understand the implication of these challenges in practice.

Other areas for future work include an understanding of the trust level provided by the public ledger in relation to the number of monitors. A large number of monitors will improve the trust level. However, because of the human-in-the-loop, this method may not scale. A potential solution may involve using V2V (Vehicle to Vehicle) technology. This would hinge on the adoption of the proposed approach by vehicle OEMs (Original Equipment Manufacturer) and will require new policies.

Finally, because of the large number of shipments and their varying nature (distance, load sharing, trans-shipment), the public ledger may grow rapidly. Being able to prune the ledger intelligently in order to manage its size is another open area of research. It is possible to consider a set of geographically delineated public ledgers with a suitable hand-off mechanism from one ledger to another.

Author Contributions: All authors contributed to the conceptual design of the framework. Haoyan Wu and Zhijie Li performed the implementation and testing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Harrison, A.; van Hoek, R.I. *Logistics Management and Strategy*; Pearson Education: London, UK, 2008.
2. Xu, X.; Weber, I.; Staples, M.; Zhu, L.; Bosch, J.; Bass, L.; Pautasso, C.; Rimba, P. A Taxonomy of Blockchain-Based Systems for Architecture Design. In Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA), Gothenburg, Sweden, 3–7 April 2017; pp. 243–252.
3. Lundbaek, L.N.; Huth, M. Oligarchic Control of Business-to-Business Blockchains. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Paris, France, 26–28 April 2017; pp. 68–71.
4. Schollmeier, R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In Proceedings of the IEEE First International Conference on Peer-To-Peer Computing, Linköping, Sweden, 27–29 August 2001; pp. 101–102.
5. Ripeanu, M. Peer-to-peer architecture case study: Gnutella network. In Proceedings of the IEEE First International Conference on Peer-To-Peer Computing, Linköping, Sweden, 27–29 August 2001; pp. 99–100.
6. Iacovou, C.L.; Benbasat, I.; Dexter, A.S. Electronic data interchange and small organizations: Adoption and impact of technology. *Manag. Inf. Syst. Q.* **1995**, *19*, 465–485.
7. Pezoa, F.; Reutter, J.L.; Suarez, F.; Ugarte, M.; Vrgoč, D. Foundations of JSON schema. In Proceedings of the 25th International Conference on World Wide Web, Montréal, QC, Canada, 11–15 April 2016; pp. 263–273.
8. Beck, R. Supply Chain Segmentation: The Next Step in Supply Chain Excellence. Technical Report. Available online: http://info.e2open.com/rs/e2open/images/WP_SC_Segmentation.pdf (accessed on 1 November 2017).
9. Dickersbach, J.T. *Supply Chain Management with APO: Structures, Modelling Approaches and Implementation of MySAP SCM 4.1*; Springer Science & Business Media: Berlin, Germany, 2005.
10. Daithankar, J.; Pandit, T. *Transportation Management with SAP TM 9: A Hands-On Guide to Configuring, Implementing, and Optimizing SAP TM*; Apress: New York, NY, USA, 2014.

11. Parfett, M. *What Is EDI? A Guide to Electronic Data Interchange*; Blackwell Publishing: Hoboken, NJ, USA, 1992.
12. Qureshi, K.N.; Abdullah, A.H. A survey on intelligent transportation systems. *Middle-East J. Sci. Res.* **2013**, *15*, 629–642.
13. Rezaei, J.; Ortt, R.; Trott, P. How SMEs can benefit from supply chain partnerships. *Int. J. Prod. Res.* **2015**, *53*, 1527–1543.
14. Kumar, R.; Kumar Singh, R. Coordination and responsiveness issues in SME supply chains: A review. *Benchmark. Int. J.* **2017**, *24*, 635–650.
15. Nakamoto, S. Bitcoin: A Peer-To-Peer Electronic Cash System. Available online: <https://bitcoin.org/bitcoin.pdf> (accessed on 1 November 2017)
16. Yli-Huumo, J.; Ko, D.; Choi, S.; Park, S.; Smolander, K. Where Is Current Research on Blockchain Technology?—A Systematic Review. *PLoS ONE* **2016**, *11*, e0163477.
17. Centobelli, P.; Cerchione, R.; Esposito, E. Environmental sustainability in the service industry of transportation and logistics service providers: Systematic literature review and research directions. *Trans. Res. Part D Trans. Environ.* **2017**, *53*, 454–470.
18. Correia, E.; Carvalho, H.; Azevedo, S.G.; Govindan, K. Maturity Models in Supply Chain Sustainability: A Systematic Literature Review. *Sustainability* **2017**, *9*, 64.
19. Andrade, N.; Mowbray, M.; Lima, A.; Wagner, G.; Ripeanu, M. Influences on cooperation in bittorrent communities. In Proceedings of the 2005 ACM SIGCOMM Workshop on Economics of Peer-To-Peer Systems, Philadelphia, PA, USA, 22–26 August 2005, pp. 111–115.
20. Fox, G. Peer-to-peer networks. *Comput. Sci. Eng.* **2001**, *3*, 75–77.
21. Guha, S.; Daswani, N. *An Experimental Study of the Skype Peer-To-Peer Voip System*; Technical Report; Cornell University Press: Ithaca, NY, USA, 2005.
22. IBM Institute for Business Value. *Next Generation Supply Chain powered by Cognitive and Blockchain*; Technical Report, Digital Supply Chain; International Business Machines Corporation: Armonk, NY, USA, 2017.
23. Smith, J. Blockfreight: Blockchain Technology for Global Freight. Technical Report. Available online: <https://bravenewcoin.com/assets/Whitepapers/BlockfreightWhitepaperFinalDraft.pdf> (accessed on 1 November 2017).
24. Toyoda, K.; Mathiopoulou, P.T.; Sasase, I.; Ohtsuki, T. A Novel Blockchain-Based Product Ownership Management System (POMS) for Anti-Counterfeits in The Post Supply Chain. *IEEE Access* **2017**, doi:10.1109/ACCESS.2017.2720760.
25. Nakasumi, M. Information Sharing for Supply Chain Management Based on Block Chain Technology. In Proceedings of the 2017 IEEE 19th Conference on Business Informatics (CBI), Thessaloniki, Greece, 24–27 July 2017; Volume 1, pp. 140–149.
26. Kshetri, N. Can Blockchain Strengthen the Internet of Things? *IT Prof.* **2017**, *19*, 68–72.
27. Bocek, T.; Rodrigues, B.B.; Strasser, T.; Stiller, B. Blockchains everywhere—a use-case of blockchains in the pharma supply-chain. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 May 2017; pp. 772–777.
28. Tian, F. An agri-food supply chain traceability system for China based on RFID & blockchain technology. In Proceedings of the 2016 13th International Conference on IEEE Service Systems and Service Management (ICSSSM), Kunming, China, 24–26 June 2016; pp. 1–6.
29. Xia, Q.; Sifah, E.B.; Smahi, A.; Amofa, S.; Zhang, X. BBDS: Blockchain-Based Data Sharing for Electronic Medical Records in Cloud Environments. *Information* **2017**, *8*, 44.
30. Zheng, Z.; Xie, S.; Dai, H.N.; Wang, H. *Blockchain Challenges and Opportunities: A Survey*; Work Paper; Inderscience Publishers: Geneva, Switzerland, 2016.
31. Zhao, J.L.; Fan, S.; Yan, J. Overview of business innovations and research opportunities in blockchain and introduction to the special issue. *Financ. Innov.* **2016**, *2*, 28.
32. Zyskind, G.; Nathan, O. Decentralizing privacy: Using blockchain to protect personal data. In Proceedings of the 2015 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 21–22 May 2015; pp. 180–184.

33. Li, Z.; Wu, H.; King, B.; Ben Miled, Z.; Wassick, J.; Tazelaar, J. On the Integration of Event-Based and Transaction-Based Architectures for Supply Chains. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, USA, 5–8 June 2017; pp. 376–382.
34. Pilkington, M. Blockchain technology: Principles and applications. In *Research Handbook on Digital Transformations*; Edward Elgar Publishing: Northampton, MA, USA, 2015.
35. Pike, R. *The Go Programming Language*; Addison-Wesley: Boston, MA, USA, 2015.
36. Wei-ping, Z.; Ming-Xin, L.; Huan, C. Using MongoDB to implement textbook management system instead of MySQL. In Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN), Xi'an, China, 27–29 May 2011; pp. 303–305.



© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).