Chapter 8

MODELING LIABILITY DATA COLLECTION SYSTEMS FOR INTELLIGENT TRANSPORTATION INFRASTRUCTURE USING HYPERLEDGER FABRIC

Luis Cintron, Scott Graham, Douglas Hodson and Barry Mullins

Abstract Distributed ledger technology is transforming environments where the participating entities have low trust. Employing distributed ledgers for intelligent transportation infrastructure communications and operations enables decentralized collaboration between entities that do not fully trust each other. This chapter models a transportation event data collection system as a Hyperledger Fabric blockchain network and simulates it using a transportation environment modeling tool. Data structures model the data collected about accidents involving vehicles and witness reports from nearby vehicles and road-side units that observed the events. The chaincode developed for the collection, validation and corroboration of the reported data is presented. Network performance results for various configurations are discussed. Optimization of the network configuration parameters resulted in a 48.1% improvement in transaction throughput. The experiments demonstrate that a distributed ledger technology such as Hyperledger Fabric holds promise for the collection of transportation data and the collaboration of applications and services that consume the data.

Keywords: Intelligent transportation infrastructure, distributed ledger, blockchain

1. Introduction

Intelligent transportation systems are information-intensive tools that facilitate connected, integrated and automated transportation systems in modern transportation infrastructures [28]. Intelligent transportation systems enable vehicles, pedestrians and infrastructure components to communicate and in-

Published by Springer Nature Switzerland AG 2019
J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XIII, IFIP AICT 570, pp. 137–156, 2019.
https://doi.org/10.1007/978-3-030-34647-8_8

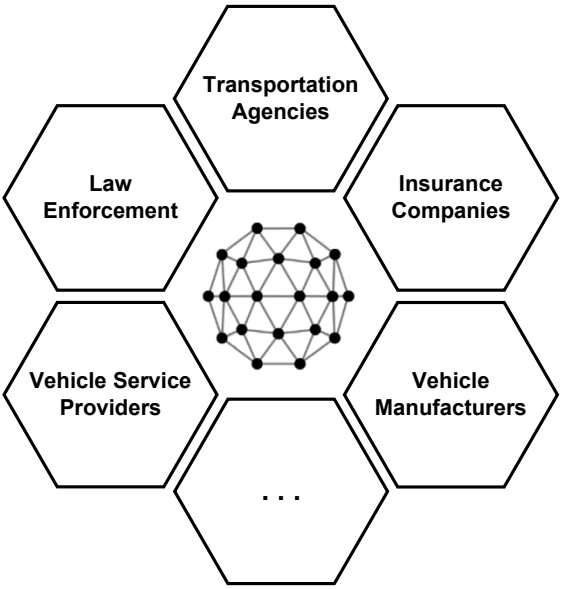

Figure 1. Transportation infrastructure consortium members.

teract with each another and to provide services to infrastructure stakeholders such as government entities and businesses. However, these systems are often isolated and do not communicate with each other due to the lack of network services, ownership/control (company-owned and maintained) and geopolitical limitations (cities, states and countries).

Distributed ledger technology, which has changed the way transactions are conducted in environments with limited or zero trust among peers, can enhance the collection, sharing and storage of data among intelligent transportation systems by providing decentralized collaborative platforms for stakeholders. Indeed, embedding distributed ledger technology in the intelligence transportation ecosystem can address communications and interoperability challenges while providing governance, security and privacy benefits that are not currently available in transportation infrastructures and services.

This chapter describes an approach for standing up a distributed ledger network (DLN) infrastructure that significantly enhances accident event data collection in an intelligent transportation infrastructure. The proposed approach involves standing up a consortium-based distributed ledger network infrastructure to serve as the back-end for multiple intelligent transportation system applications within the same instance. The decentralized network infrastructure, which is developed using the Hyperledger Fabric framework and Hyperledger Composer toolset, is designed to be operated and maintained by a consortium of government and non-government entities such as law enforce-

ment, transportation agencies, insurance companies, vehicle manufacturers and other transportation-related service providers (Figure 1). The infrastructure supports the integration, collaboration and maintenance of relevant data by pre-selected parties in a decentralized and secure manner.

The focus is on a trusted, secure and verifiable repository for data collected by vehicles, infrastructure components and participants, which would extend the accident reconstruction work of Kopylova et al. [10]. Specifically, the distributed ledger network infrastructure would serve as a platform for executing distributed network services for reconstructing events leading up to, during and immediately following vehicle accidents. Sensors mounted on vehicles and road-side units (RSUs) collect data about vehicle parameters (e.g., speed, heading, location) as well as data about other vehicles shared via vehicular ad-hoc network (VANET) vehicle-to-vehicle/vehicle-to-infrastructure communications using the IEEE 1609 family of standards for wireless access in vehicular environments (WAVE). This capability would enhance vehicle forensics and improve processes and tools for identifying the root causes of accidents and the liable parties.

2. Background

Intelligent transportation systems comprise devices and sensors that collect, transmit and analyze data and information to provide services that enhance the quality of and experiences provided by modern transportation infrastructures [25].

The data collection components in intelligent transportation systems include sensors such as cameras, GPS receivers, RFID readers and radar systems that are embedded in vehicles and road-side units. The continuous collection and analysis of observed data enables vehicles to detect and avoid collisions, report traffic conditions and pass on other information witnessed during road events such as accidents. Systems and sensors may share the collected data with other vehicles, road-side units and remote services via proximal vehicular ad-hoc networks or through network communications technologies such as Ethernet and 3G/4G/5G. The transmitted data is analyzed and processed to provide services such as congestion control, automatic toll collection and collision prevention, among others. In other words, intelligent transportation systems rely on information collection and dissemination to provide services to transportation infrastructure stakeholders.

Currently, communications between intelligent transportation systems are hindered by the high cost of operation and maintenance, lack of network capabilities, issues of data ownership/control (company-owned and maintained) and geopolitical limitations (imposed by jurisdictions such as cities, states and countries). As a result, the search for innovative and cost-saving solutions to create a connected ecosystem of intelligent transportation systems is an active area of research in large cities such as New York City [17] and Tampa [26].

Distributed ledger technology enables the maintenance of append-only data structures by untrusted or partially-trusted participants in a decentralized

manner [3]. It leverages protocols that provide decentralized communications, tamper-resistant storage of transactions, crash/fault tolerance, and data provenance, as well as other features such as code execution via smart-contracts or chaincode. The resulting deployments are often referred to as distributed ledger networks.

Popular distributed ledger networks, which utilize blockchains [16] or directed acyclic graphs [2] to maintain ledgers, are categorized based on the level of trust required for peers to participate. Public or permissionless distributed ledger networks are accessible to anyone on the Internet and their contents are visible and verifiable by all participants [21]. Access permissions in private or permissioned distributed ledger networks are maintained by single central entities where the network peers are highly trusted. Consortium chains, as described in [5], are partially-decentralized solutions that are hybrids of low-trust (i.e., public blockchains) and single high-trust entity models (i.e., private blockchains) [21]. Such hybrid models enable organizations in a consortium to share transaction records without having to trust all the other organizations in the network or rely on a trusted third party to facilitate communications [9]. The consortium-based distributed ledger network model is a good fit for transportation infrastructure applications due to its privacy-enabling and access control features, distributed execution and low-cost scalability. Ideally, the consortium would comprise government entities such as law enforcement and transportation departments as well as organizations that provide services to the transportation infrastructure or its stakeholders (e.g., vehicle manufacturers, automobile dealers, insurance companies and maintenance service providers).

3. Related Work

Kopylova et al. [10] have presented an approach for collecting vehicular ad-hoc network data to reconstruct the events that took place before, during and after accidents. The approach leverages vehicles with improved logging mechanisms, vehicular ad-hoc network communications data and a GPS data rectification mechanism that processes data submitted by other entities. All the events are logged in the associated vehicle data recorders that require owner consent or court orders to gain access to the stored data. This data is parsed, filtered and appended to previously-acquired witnessed data in order to perform forensic analyses. Potential problems are that the data stored in the vehicles is at risk of tampering via modification or deletion, and the data may not always be accessible.

Dorri et al. [8] have proposed the use of a blockchain-based distributed ledger network to address scalability issues with centralized systems (e.g., cloud services), preserve the privacy of vehicle owners and passengers, and enhance the security of smart transportation systems. Unlike permissionless public blockchains such as Bitcoin, the approach clusters the network and moves distributed ledger network management to nodes whose sole purpose is to broadcast and verify transactions and append blocks to the ledgers. Important features include hash checks of wireless software updates, secure data exchange

with insurance providers and car-sharing services. Security mechanisms in the blockchain design include a chain of block hashes, encryption of transactions and public-key-based authentication of transactions. The distributed network architecture prevents service disruptions caused by distributed denial-of-service attacks by filtering transactions from entities with invalid keys. Unfortunately, Dorri and colleagues have not conducted any experimentation of their approach, even in a simulated transportation environment.

Oham et al. [19] have described a blockchain liability attribution framework for autonomous vehicles based on a consortium of transportation and government organizations. The framework employs two partitions for communications – operational and decision partitions – that collect and share data between different entities and sensors. A qualitative analysis of the framework demonstrates its resilience to malicious activities such as transaction deletion, collusion and spoofing. A performance evaluation focusing on the average verification and validation times for different types of transactions is promising; the results show less overhead compared with the approach of Cebe et al. [7].

In other work, Oham and colleagues [18] have developed a blockchain framework for auto-insurance claims and adjudication for connected and automated vehicles. However, both works by Oham et al. [18, 19] are unclear about how the consensus algorithm operates to ensure network integrity and they omit design considerations for enabling the services to operate within existing intelligent transportation systems. Unlike the other proposals described in this section [7, 18, 19], the implementation described in this chapter leverages an open-source framework that has been tested in production environments, has community and commercial support and continuing upgrades while enabling a number of applications within a single platform.

While previous research describes implementations of distributed ledger networks that enhance intelligent transportation applications, little, if any, work has focused on modeling transportation infrastructure applications or intelligent transportation systems that leverage distributed ledger network frameworks. Additionally, evidence showing how distributed ledger networks can scale to millions of vehicles in public roads and in other transportation environments is minimal. In particular, scalability challenges related to consensus algorithms, performance metrics, and designs and decisions related to operational distributed ledger networks for intelligent transportation systems have not been discussed. Furthermore, the other approaches described in this section have not been analyzed in terms of key parameters such as block size, block timeout and transactions per block that affect overall performance parameters such as transaction throughput and consensus time, which are vital in intelligent transportation ecosystems.

4. Infrastructure Modeling and Implementation

The goal of this work was to model an intelligent transportation infrastructure and applications using a distributed ledger network, specifically, Hyperledger Fabric. Performance metrics were recorded and analyzed to assess the

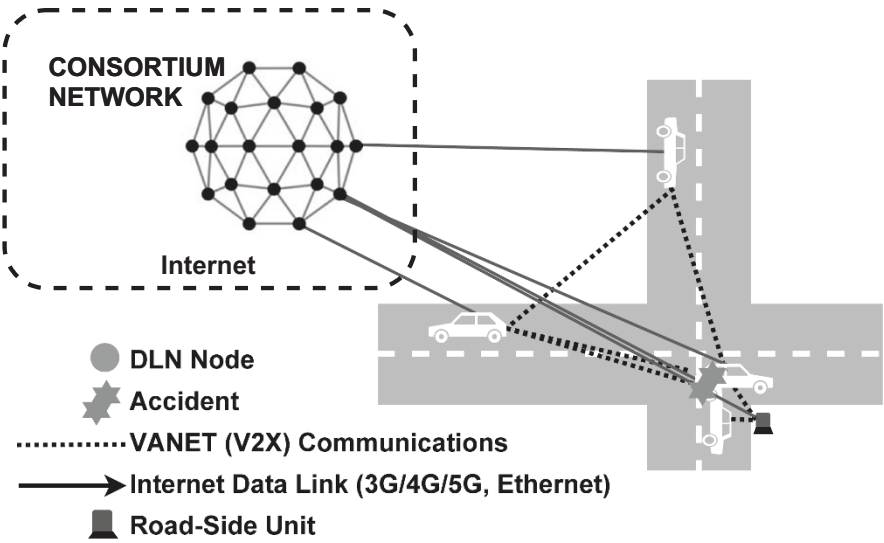

Figure 2. Operational view of an infrastructure with a consortium network.

effectiveness of the network at handling the data volumes encountered in an active intelligent transportation infrastructure. Network and code design parameters that improve responsiveness and throughput are also discussed. Figure 2 shows the operational view of the modeled infrastructure, which incorporates vehicles and road-side units, and their interactions with the consortium network.

4.1 Definitions

The following are the key terms used in the model:

- **Consensus:** Consensus refers to agreement in a distributed ledger network about the next set of transactions and the order in which they are appended to the ledger [15]. Consensus in Hyperledger Fabric takes place among ordering service nodes (commonly referred to as orderers) and is achieved by selecting a leader from among the nodes with a fully synchronized ledger to order the transactions, place them in a block and deliver them to other peer nodes for validation and committal. Apache Kafka was employed for consensus because it is the only implementation provided by Hyperledger Fabric (v1.2) that is suitable for production environments.

- **Channel:** A channel in Hyperledger Fabric is a private blockchain that only channel participants can access and interact with [15]. Participation is managed via authentication and access control policies.

- **Chaincode:** Chaincode is the code/service invoked by an application that interacts with the Hyperledger Fabric network to manage accesses and modifications to the ledger. It is installed on peer nodes to work on one or more available channels [15].
- **Endorsement:** Endorsement in Hyperledger Fabric is the simulation of the execution of a chaincode transaction by a peer node and the communication of the response back to the originator along with the peer node signature to provide proof of a valid execution result [15]. Endorsement policies specify transaction endorsement requirements using Boolean expressions involving the participating organizations [27].
- **Membership Services Provider:** A membership services provider (MSP) supplies cryptographic (public-key infrastructure based) credentials to Hyperledger Fabric participants for authentication and transaction processing [15].
- **Peer Node:** A peer node is a network node that executes the chaincode and maintains a copy of the ledger. Peer nodes designated as endorsers can participate in the endorsements of transactions. Nodes can also be designated as anchors, which enables them to be discovered by and communicate with all the other peer nodes.

4.2 Implementation Platform

The network node simulations were executed in virtual machines (VMs) on a single workstation powered by an Intel CORE i7 vPro (7th Gen) processor (2.9 GHz, four cores, eight logical processors) with 16 GB of RAM. Each Hyperledger Fabric virtual machine node was allocated two logical processors, 2 GB RAM and ran the Ubuntu 16.04 64-bit operating system.

4.3 Frameworks and Tools

Several frameworks and tools, which are part of the Hyperledger collaborative effort hosted by the Linux Foundation, were employed. These frameworks and tools are maintained by technology leaders such as IBM, Intel and SAP. In particular, the following frameworks and tools were used in this research:

- **Hyperledger Fabric:** Hyperledger Fabric (v1.2) is a modular and extensible open-source platform for deploying and operating permissioned distributed ledgers; it is hosted by the Linux Foundation and maintained by IBM [1]. Its modularity enables architects and developers to tailor various layers such as methods for validation, consensus and distributed ledger data structures to meet an organization's needs. Furthermore, Hyperledger Fabric supports the creation of a consortium-based network of peers in which organizations can manage their own user permissions. Hyperledger Fabric served as the distributed ledger network backbone for this research.

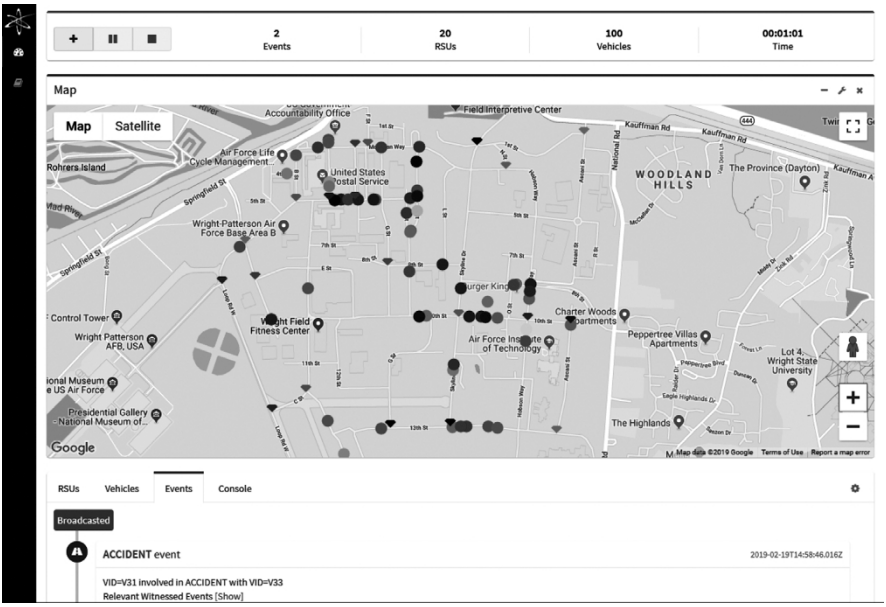

Figure 3. AFIT Lightweight Transportation Modeling Tool user interface

- **Hyperledger Composer:** Hyperledger Composer (v0.20.3) is a toolset that supports the development and execution of blockchain networks and services [13]. It was employed to model and deploy intelligent transportation infrastructure chaincode and network services. Hyperledger Composer incorporates the Playground tool for viewing and interacting with world-state data and performing upgrades to services. It can also spawn a REST server that interfaces with the Hyperledger Fabric network to provide a web application programming interface.

Vehicles, road-side units, and infrastructure behavior and communications were modeled using a custom tool:

- **AFIT Lightweight Transportation Modeling Tool:** This tool provides an intuitive environment for modeling and simulating vehicle movements and their communications with other vehicles and entities in a transportation infrastructure. The tool, developed using JavaScript and NodeJS, can be deployed as a web application or as a standalone application. In the experiments, the tool was used to generate vehicular traffic and accident scenarios, and to function as an application client that interacted with the distributed ledger network (Figure 3).

Table 1. Baseline Hyperledger Fabric network configuration.

Parameter	**Value**
Participating Organizations	3
Orderer Nodes	3
Peer Nodes	3
Channels	1
Zookeeper-Kafka Cluster Nodes	3 Zookeeper, 4 Kafka
World-State Database	CouchDB
Block Size	99 MB or 10 Tx/Block
Block Timeout	2 s
Endorsement Policies	ORG1, ORG2, ORG3

4.4 Experimental Network

Table 1 shows the baseline configuration of the Hyperledger Fabric network with a Zookeeper-Kafka node cluster. Figure 4 shows the network topology corresponding to this configuration. Membership services providers and peer containers for each organization were instantiated in the same virtual machine. The peer virtual machines executed the Composer Rest Server to expose the web application programming interface. All the virtual machines were configured with static IP addresses and the Docker container configurations were set to the host network mode.

The number of participating organizations (consortium members) was chosen to be three because it is the minimum number of organizations required to create a partial-trust environment where a blockchain or distributed ledger network would be most beneficial (this is not enforced by Hyperledger Fabric). There are other more efficient ways than a distributed ledger for storing and managing data for a single organization; one possibility is a distributed database. When two organizations collaborate, the collaboration assumes full trust between the organizations and there is no need for endorsement or consensus with regard to data distribution. However, in a consortium of three organizations, the possibility exists that not all the organizations would trust each other, increasing the importance of endorsement, and leader-based or voting-based consensus.

The seven nodes used in the Zookeeper-Kafka cluster configuration is the minimum number needed for Zookeeper-Kafka consensus in Hyperledger Fabric (v1.2). In the case of the Zookeeper nodes, the number should be odd to avoid split-brain scenarios and should be larger than one to avoid a single point of failure [12]; hence, the configuration employed three Zookeeper nodes. In the case of the Kafka nodes, four nodes is the minimum number needed to exhibit crash/fault tolerance [12].

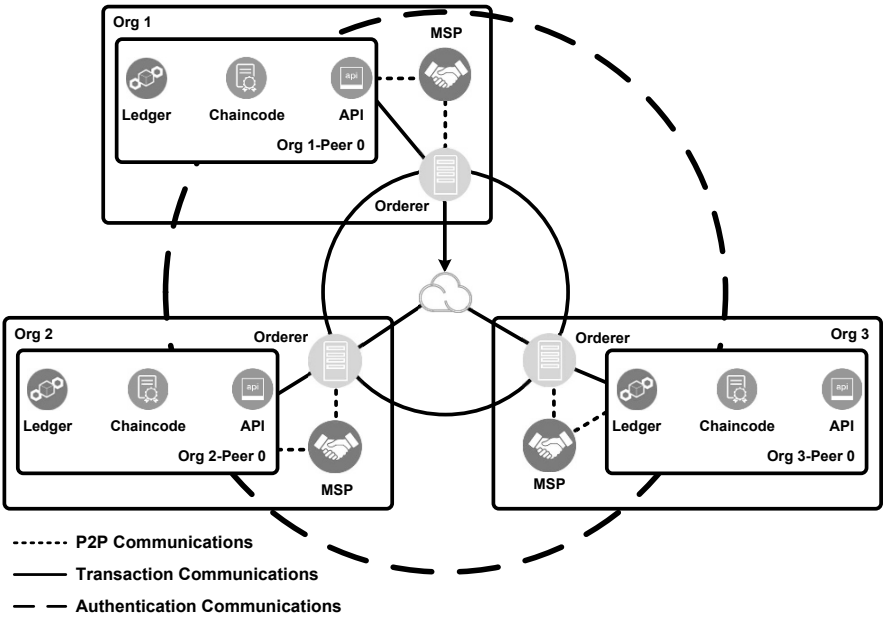

Figure 4. Experimental Hyperledger Fabric network configuration.

4.5 Assumptions

The following assumptions were made in the experiments:

- **Transaction Latency:** Transaction times were measured from the time the client submits a transaction to the network for validation and consensus to the time the corresponding block is created and broadcasted.
- **Signal Loss:** Wireless and wired signal losses were assumed to be minimal and were not modeled in the simulation environment.
- **Vehicle and Sensor Authentication:** Vehicles and road-side units that send or manage transactions were assumed to have appropriate access rights to the services.
- **Traffic Laws:** Vehicles were assumed to not stop at intersections and not adjust their speeds based on roadway speed limits. Also, vehicular traffic lanes were not considered.
- **Obstacles:** Aside from other vehicles in the same path, no other obstacles (e.g., pedestrians or animals) were assumed to exist.
- **Event Data Recording:** Vehicles were assumed to have event data recorders that stored times, vehicle IDs, locations, speeds, headings and

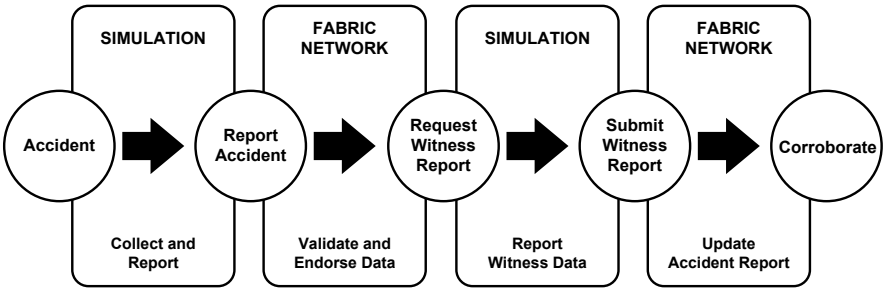

Figure 5. Application event workflow.

misbehavior data of vehicles in range. This capability is typically enabled via on-board units.

- **Vehicle Communications:** Vehicles were assumed to be equipped with the means to establish zero-latency data links with other vehicles and road-side units, and could communicate with Hyperledger Fabric services over the Internet. The inner workings of vehicular network communications as specified by the IEEE 1609 family of standards were not modeled.

5. Accident Data Collection

This research focused on the storage of accident event reports and witnessed data recorded by vehicles and road-side units in order to create snapshots of events within a window of time before, during and after an accident, providing evidence that could identify the liable parties. Each vehicle in the simulation shared information with other vehicles and the infrastructure via vehicle-to-vehicle or vehicle-to-infrastructure communications channels, and with the transportation distributed ledger network (Figure 4) via connections to the Internet (Figure 2). Vehicles were equipped with event data recorders that logged their sensor data as well as sensor data received from other vehicles. The broadcasted messages contained GPS position, heading and current speed as in other implementations [10]. The vehicular *ad hoc* network parameters, which were based on beacon data in [10], comprised vehicle ID, location, speed and heading.

5.1 Scenario Generation

Each simulation scenario involved a predefined number of vehicles and road-side units in a specified area. Each vehicle was assigned an origin and destination, and the vehicle moved until it arrived at the destination. Only road-side units were assumed to collect data about misbehaving vehicles. Vehicles broadcasted their parameters to other vehicles within a range of 100 m.

Table 2. Modeled data and attributes.

Name	Type	Attributes
Sensor	asset	id, type
Vehicle extends Sensor	asset	odometer, eventsInvolved
RSU extends Sensor	asset	location
RoadEvent	asset	id, location, eventtimestamp, type, vehiclesInvolved, witnessedData, validated, sourceSensor
RoadEventTx	transaction	Same as RoadEvent
WitnessedData	asset	id, observedVehicle, sourceSensor, roadEventId, location, eventtime, speed, heading, distanceFromsource, behavior, nearbySensors
WitnessDataTx	transaction	Same as WitnessedData

The simulation triggered an accident when two or more vehicles were within the collision distance. At this point, one of the involved vehicles reported the event to the distributed ledger network along with its logged data and the IDs of the other involved vehicles. The distributed ledger network then validated the source vehicle and the involved vehicles, created an accident event report and notified the simulation application after the report was submitted. Next, the simulation application notified all the vehicles in the area about the accident and requested them to report witnessed data within the accident location during the time frame of the accident. Road-side units were also informed about the accident and were requested to report witnessed misbehavior data. Figure 5 shows the event workflow in the experiments.

5.2 Network Data Models

The models used to store data about road events were defined using Hyperledger Composer. Hyperledger Composer employs its own object-oriented modeling language, Composer Modeling Language (CML). Table 2 presents the model definitions. Vehicle and road-side unit owners were not modeled in the experiments, but they could easily be included in an operational environment. Note that the only Composer Modeling Language reference in the initial model definitions is between Sensor and RoadEvent. The received witnessed data was appended to the RoadEvent.WitnessedData collection. This design raised some database concurrency issues, which are discussed below.

5.3 Chaincode

Chaincode was packaged and deployed using Hyperledger Composer. In Hyperledger Composer, chaincode logic is developed using JavaScript in the form of transaction processor functions and it is part of the business network

Algorithm 1 : Submit a transaction for a road event.

```
tx ← {sourceId, eventId,time,location,VehiclesInvolved,WitnessedData}
if SensorExists(tx.sourceId) AND IsValid(tx) then
    re ← RoadEvent(tx)
    re.source ← SensorAssetRegistry.get(tx.sourceId, sensorType)
    EventAssetRegistry.add(re)
    for i = 0 to tx.VehiclesInvolved.length do
        //create record of vehicle observed to be involved in event
        v ← SensorAssetRegistry.get(tx.VehiclesInvolved[i])
        v.eventsInvolved.add(re)
        SensorAssetRegistry.update(v)
    end for
    emit(RoadEventSubmitted)
else
    emit(InvalidSourceVehicleEvent)
end if
```

Algorithm 2 : Submit a witnessed data transaction for a road event.

```
tx ← {sourceId, eventId, WitnessedData}
if tx.WitnessedData.length > 0 AND SensorExists(tx.sourceId) AND
RoadEventExists(tx.eventId) AND IsValid(tx) then
    wd ← WitnessedData(tx)
    WitnessedDataRegistry.add(wd)
    emit(WitnessedDataSubmitted)
else
    emit(InvalidWitnessedDataTx)
end if
```

archive that is deployed to the Hyperledger Fabric network to provide capabilities and services. Transaction processor functions are automatically invoked when transactions are submitted from the application programming interfaces generated by Hyperledger Composer. Additionally, transaction processor functions reference the data models and describe how to use transaction objects to create road event reports and append the witnessed data to the reports. The procedures defined in chaincode must be passed arguments (if required) that are objects of transaction type classes listed in Table 2. The chaincode must validate the existence of sensors that submit transactions and also validate data (e.g., bounds validation) before creating an asset.

Algorithm 1 specifies how a road event report is created.

Algorithm 2 specifies how a road event is updated with witnessed data.

Algorithm 3 specifies how the network validates witnessed data about an event to obtain consensus that the event did indeed occur (i.e., same observed behavior by multiple unrelated parties), and help identify potential misbehavior. This transaction processor function can be triggered by a consortium entity (e.g., insurance company or law enforcement) looking into an event or it could

Algorithm 3 : Corroborate witnessed data for a specific road event.

```
re ← {RoadEvent}
for all WitnessReport observed ∈ RoadEvent re do
    possibleValidators ← getWitnessReports(ΔT, observed, WitnessReports ∈ re)
    for all WitnessReport p ∈ possibleValidators do
        if p validates observed then
            observed.validatedBy ← observed.validatedBy ∪ p.id
        end if
        if p.seenInRange(observed) then
            observed.seenBy ← observed.seenBy ∪ p.id
        end if
    end for
end for
validated ← getValidatedReports()
seen ← getSeenVehicleReports()
possibleSpoofers ← re.WitnessReports − (validated ∩ seen)
```

be triggered automatically after a specified period of time. After the chaincode is executed on a transaction, the transaction awaits endorsement, following which it is sent for ordering.

5.4 Analysis of Data

During the initial tests, a number of unsuccessful transactions occurred when submitting the witnessed data reports and when performing stress tests with repeated vehicle IDs. The unsuccessful transactions reported "Error trying invoke chaincode" and "Error: Peer has rejected transaction with code MVCC_READ_CONFLICT." This is due to the multi-version concurrency control employed by Hyperledger Fabric, which requires the state of an object to be read or written during the commit phase to be the same as when the transaction was endorsed during the execution phase [4, 20, 27]. The errors were caused by fast update rates of road event assets with witnessed data from all the other entities. They were eliminated by creating a new object for each witnessed data transaction. The solution employs unique IDs to reference the road event and sensors in a report, an approach analogous to the use of foreign keys in relational databases.

Scenarios with ten to 100 road events in increments of ten were submitted within a one second time period. All the submissions were distributed over the network peers in a round-robin manner. Figure 6 shows the network performance with the baseline configuration. A peak throughput of 10.0 transactions per second (TPS) and an average of 8.82 transactions per second were measured in the application layer.

The block size and timeout parameters, and the endorsement policies were modified based on the optimization recommendations in [27]. The network configuration shown in Table 3 resulted in the best overall performance.

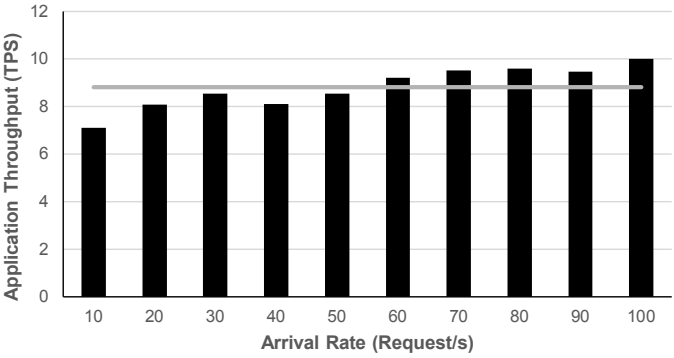

Figure 6. Network performance with the baseline configuration.

Table 3. Optimized Hyperledger Fabric network configuration.

Parameter	**Value**
Block Size	99 MB or 100 Tx/Block
Block Timeout	1 s
Endorsement Policies	Two of ORG1, ORG2, ORG3

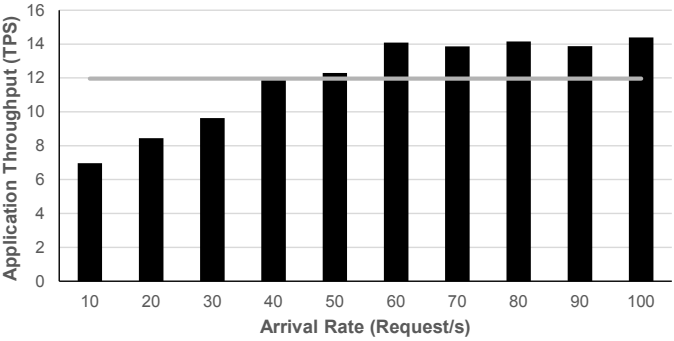

Figure 7. Network performance with the optimized configuration.

Figure 7 shows the network performance with the optimized configuration. A peak throughput of 14.4 transactions per second was measured in the application layer, a 48.1% improvement over the baseline configuration. Moreover, the average response time was reduced by 33.4%. Increasing the transaction arrival rate over the throughput limit over long periods of time often resulted in

network timeout errors or unresponsive servers, essentially distributed denial-of-service.

6. Discussion

The implementation of an intelligent transportation infrastructure data collection system using Hyperledger Fabric has benefits, drawbacks and challenges; it also raises some privacy and security issues:

6.1 Benefits

A key benefit of using a blockchain network is the access to a pseudo-immutable ledger that records transactions and world-state changes in a cryptographically-secure manner. The network provides native auditing services that could ensure the reliability and safety of modern transportation infrastructures.

The modularity of Hyperledger Fabric makes it an attractive framework for implementing a distributed ledger network that can interface with an infrastructure and services. By resolving transactions into a world-state database, stakeholders can execute rich queries to obtain data if they have the proper access rights. Additionally, as a permissioned network framework, its implementation ensures zero participant anonymity because all the identities in the distributed ledger network are authenticated. As a result, participants are always accountable for their actions (e.g., for certificate revocations, traffic violations and accident liability).

6.2 Drawbacks and Challenges

Although the implementation is crash/fault tolerant as a result of using Apache Kafka, it is not Byzantine fault tolerant. Byzantine faults refer to faulty nodes that may appear to be fully functional, but may produce inconsistent results unknowingly or maliciously. The ordering nodes can be rendered resilient to Byzantine faults by implementing a different consensus algorithm. Sousa et al. [24] have developed a Byzantine-fault-tolerant consensus module called BFT-SMART, with a tentative execution of requests approach similar to the practical Byzantine fault tolerance approach of Castro and Liskov [6]. However, this module is not included in Hyperledger Fabric and its performance and reliability in production environments are still unknown.

Storage is a concern given the large amount of data transacted in transportation environments. Since transactions recorded in the blockchain ledger cannot be erased or tampered with, the ledger grows in size quickly. As a result, peer nodes must have adequate storage to accommodate this data, which results in high hosting costs over time for all participants. Consequently, nodes that experience downtime suffer long synchronization times that could extend endorsement downtime and lead to transaction execution failures. Finally, framework components such as those provided by Hyperledger Composer are not mature enough and, therefore, suffer from reliability issues that could result

in nonresponsiveness during periods with high transaction arrival rates, as was encountered in the experiments.

Processing times for applications with high-throughput requirements are also a concern. Based on the results obtained in the experimental network, several variables have to be considered when designing a distributed ledger network that would support the processing and storage of data at high rates. In production environments, it is expected that applications relying on a Hyperledger Fabric implementation of a distributed ledger network could handle thousands of vehicles, road-side units and users conducting transactions every minute. Although Thakkar et al. [27] have demonstrated that a Hyperledger Fabric implementation can reach a throughput of 2,800 transactions per second, determining whether or not the solution is adequate for a transportation infrastructure would depend on the requirements of the applications that are deployed.

6.3 Security and Privacy Considerations

Public-key-infrastructure services and access control rules allow secure access to data by privileged users as defined by the consortium. Implementations can enable users (e.g., vehicle owners) to control access to data involving their vehicles. A vehicle public-key infrastructure (VPKI) as defined by the IEEE 1609.2 standard can be supported in a Hyperledger Fabric implementation by integrating vehicle certificate manager services in the membership services providers. Since such an infrastructure relies on providing pseudonymity to vehicles, certificates can be utilized to authenticate vehicles or sign transaction data and increase the trust in data sources without revealing their identities. Designated authorities can always obtain the real identities of vehicle pseudonyms in the case of accidents or legal investigations [22]. Hyperledger Fabric also allows for certificate revocation, preventing participants from accessing data after they have lost their credentials.

Hyperledger Fabric orderers, although not involved in the validation of transactions, could be compromised to gain access to all the transactions received and distributed by the Kafka cluster. These nodes could be compromised to intercept transactions sent and received by the ordering service. If information privacy is required (e.g., for personally-identifiable information), Hyperledger Fabric provides private data channels that create separate private ledgers between parties. Private channel transactions are not sent to an orderer; instead, hashes of the transactions and the timestamps are sent, preventing the orderer from observing transaction content.

The possibility exists that a participant could analyze the shared ledger data to discover traffic patterns, and the origins, destinations and times of vehicles. This information could reveal details such as home addresses, work locations and daily routines of vehicle owners. Therefore, participating organizations must be transparent in the way they handle the data. More importantly, all the stakeholders must be aware that participants could analyze data for purposes other than were intended. By incorporating access control policies in

Hyperledger Fabric, it is possible for vehicle owners to actively prevent certain users and organizations accessing data about their vehicles and behavior.

7. Conclusions

The work described in this chapter extends the approach of Kopylova et al. [10] by employing the Hyperledger Fabric framework and Hyperledger Composer toolset to create a distributed ledger network that provides services for storing, corroborating and querying accident event data in a decentralized and secure manner. Like other proposals [7, 8, 18, 19], the approach relies on vehicle on-board units to collect and disseminate vehicular ad-hoc network data while the vehicles are on the road. Data about accidents and other road events is pushed to the Hyperledger Fabric network, providing irrefutable evidence pertaining to the events for subsequent analyses. This distributed ledger network differs from the other proposals because of its use of the open-source Hyperledger Fabric platform, which supports the execution of multiple applications and channels, as well as seamless integration with existing transportation systems.

The design considerations and improvements discussed in this chapter ensure that the distributed ledger network can scale to handle the large volumes of transactions encountered in transportation infrastructures. Specifically, optimizing the configuration by adjusting block size, block timeout and endorsement policies provides significant performance improvements. However, as with any modern technology, the benefits come with some drawbacks, in this case, primarily privacy risks.

The views expressed in this chapter are those of the authors, and do not reflect the official policy or position of the U.S. Air Force, U.S. Department of Defense or U.S. Government. This document has been approved for public release, distribution unlimited (Case #88ABW-2018-6399).

References

[1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. Weed Cocco and J. Yellick, Hyperledger Fabric: A distributed operating system for permissioned blockchains, *Proceedings of the Thirteenth European Conference on Computer Systems*, article no. 30, 2018.

[2] L. Baird, The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance, Technical Report SWIRLDS-TR-2016-01, Swirlds, College Station, Texas, 2016.

[3] F. Bencic and I. Zarko, Distributed ledger technology: Blockchain compared to directed acyclic graph, *Proceedings of the Thirty-Eighth IEEE International Conference on Distributed Computing Systems*, pp. 1569–1570, 2018.

[4] P. Bernstein and N. Goodman, Multiversion concurrency control – Theory and algorithms, *ACM Transactions on Database Systems*, vol. 8(4), pp. 465–483, 1983.

[5] V. Buterin, On public and private blockchains, *Ethereum Foundation Blog* (`blog.ethereum.org/2015/08/07/on-public-and-private-blockchains`), August 6, 2015.

[6] M. Castro and B. Liskov, Practical Byzantine fault tolerance, *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, pp. 173–186, 1999.

[7] M. Cebe, E. Erdin, K. Akkaya, H. Aksu and S. Uluagac, Block4Forensic: An integrated lightweight blockchain framework for forensic applications of connected vehicles, *IEEE Communications*, vol. 56(10), pp. 50–57, 2018.

[8] A. Dorri, M. Steger, S. Kanhere and R. Jurdak, Blockchain: A distributed solution to automotive security and privacy, *IEEE Communications*, vol. 55(12), pp. 119–125, 2017.

[9] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang and E. Hossain, Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains, *IEEE Transactions on Industrial Informatics*, vol. 13(6), pp. 3154–3164, 2017.

[10] Y. Kopylova, C. Farkas and W. Xu, Accurate accident reconstruction in VANET, in *Data and Applications Security and Privacy XXV*, Y. Li (Ed.), Springer, Berlin Heidelberg, Germany, pp. 271–279, 2011.

[11] L. Lamport, R. Shostak and M. Pease, The Byzantine generals problem, *ACM Transactions on Programming Languages and Systems*, vol. 4(3), pp. 382–401, 1982.

[12] Linux Foundation, Bringing up a Kafka-Based Ordering Service, San Francisco, California (`hyperledger-fabric.readthedocs.io/en/release-1.2/kafka.html`), 2019.

[13] Linux Foundation, Hyperledger Composer, San Francisco, California (`hyperledger.github.io/composer/latest`), 2019.

[14] Linux Foundation, Hyperledger Explorer, San Francisco, California (`github.com/hyperledger/blockchain-explorer`), 2019.

[15] Linux Foundation, Hyperledger Fabric (1.2) Glossary, San Francisco, California (`hyperledger-fabric.readthedocs.io/en/release-1.2/glossary.html`), 2019.

[16] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System (`bitcoin.org/bitcoin.pdf`), 2008.

[17] New York City Department of Transportation, NYC Connected Vehicle Project for Safer Transportation, New York (`cvp.nyc`), 2019.

[18] C. Oham, R. Jurdak, S. Kanhere, A. Dorri and S. Jha, B-FICA: Blockchain-Based Framework for Auto-Insurance Claim and Adjudication, arXiv:1806.06169 (`arxiv.org/abs/1806.06169`), 2018.

[19] C. Oham, S. Kanhere, R. Jurdak and S. Jha, A Blockchain-Based Liability Attribution Framework for Autonomous Vehicles, arXiv: 1802.05050 (arxiv.org/abs/1802.05050), 2018.

[20] C. Papadimitriou and P. Kanellakis, On concurrency control by multiple versions, *Proceedings of the First ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 76–82, 1982.

[21] M. Pilkington, Blockchain technology: Principles and applications, in *Research Handbook on Digital Transformations*, F. Olleros and M. Zhegu (Eds.), Edward Elgar, Northampton, Massachusetts, pp. 225–246, 2016.

[22] Y. Qian, K. Lu and N. Moayeri, A secure VANET MAC protocol for DSRC applications, *Proceedings of the IEEE Global Telecommunications Conference*, 2008.

[23] M. Singh and S. Kim, Blockchain-Based Intelligent Vehicle Data Sharing Framework, arXiv:1708.09721 (arxiv.org/abs/1708.09721), 2017.

[24] J. Sousa, A. Bessani and M. Vukolic, A Byzantine fault-tolerant ordering service for the Hyperledger Fabric blockchain platform, *Proceedings of the Forty-Eighth Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 51–58, 2018.

[25] A. Sumalee and H. Ho, Smarter and more connected: Future intelligent transportation systems, *IATSS Research*, vol. 42(2), pp. 67–71, 2018.

[26] Tampa Hillsborough Expressway Authority, THEA Connected Vehicle Pilot, Tampa, Florida (www.tampacvpilot.com), 2019.

[27] P. Thakkar, S. Nathan and B. Viswanathan, Performance benchmarking and optimizing the Hyperledger Fabric blockchain platform, *Proceedings of the Twenty-Sixth IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 264–276, 2018.

[28] U.S. Department of Transportation, ITS Strategic Plan 2015-2019, FHWA-JPO-14-145, Washington, DC (rosap.ntl.bts.gov/view/dot/3506), 2014.

[29] Y. Yuan and F. Wang, Towards blockchain-based intelligent transportation systems, *Proceedings of the Nineteenth IEEE International Conference on Intelligent Transportation Systems*, pp. 2663–2668, 2016.