3-22-2019

Modeling a Consortium-based Distributed Ledger Network with Applications for Intelligent Transportation Infrastructure

Luis A. Cintron

MODELING A CONSORTIUM-BASED DISTRIBUTED LEDGER NETWORK WITH APPLICATIONS FOR INTELLIGENT TRANSPORTATION INFRASTRUCTURE

THESIS

Luis A. Cintron, Captain, USAF

AFIT-ENG-MS-18-M-019

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-18-M-019

MODELING A CONSORTIUM-BASED DISTRIBUTED LEDGER NETWORK WITH APPLICATIONS FOR INTELLIGENT TRANSPORTATION INFRASTRUCTURE

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Luis A. Cintron, B.S.
Captain, USAF

March 2019

AFIT-ENG-MS-18-M-019

MODELING A CONSORTIUM-BASED DISTRIBUTED LEDGER NETWORK WITH APPLICATIONS FOR INTELLIGENT TRANSPORTATION INFRASTRUCTURE

THESIS

Luis A. Cintron, B.S.
Captain, USAF

Committee Membership:

Scott Graham, Ph.D.
Chair

Barry Mullins, Ph.D., P.E.
Member

Douglas Hodson, Ph.D.
Member

AFIT-ENG-MS-18-M-019

Abstract

Emerging distributed-ledger networks are changing the landscape for environments of low trust among participating entities. Implementing such technologies in transportation infrastructure communications and operations would enable, in a secure fashion, decentralized collaboration among entities who do not fully trust each other. This work models a transportation records and events data collection system enabled by a Hyperledger Fabric blockchain network and simulated using a transportation environment modeling tool. A distributed vehicle records management use case is shown with the capability to detect and prevent unauthorized vehicle odometer tampering. Another use case studied is that of vehicular data collected during the event of an accident. It relies on broadcast data collected from the Vehicle Ad-hoc Network (VANET) and submitted as witness reports from nearby vehicles or road-side units who observed the event taking place or detected misbehaving activity by vehicles involved in the accident. Mechanisms for the collection, validation, and corroboration of the reported data which may prove crucial for vehicle accident forensics are described and their implementation is discussed. A performance analysis of the network under various loads is conducted with results suggesting that tailored endorsement policies are an effective mechanism to improve overall network throughput for a given channel. The experimental testbed shows that Hyperledger Fabric and other distributed ledger technologies hold promise for the collection of transportation data and the collaboration of applications and services that consume it.

This work is dedicated to my wife and son for their support, encouragement, and love.

Acknowledgements

I would first like to thank my research advisor Dr. Scott Graham for his continuous support and expert advice during my stay at AFIT. He allowed this thesis to be my own work while providing direction whenever he thought I needed it.

I would also like to acknowledge professors and committee members Dr. Douglas Hodson and Dr. Barry Mullins for their helpful lessons, advice during the research process, and their thorough reviews of the work I produced.

Finally, I must express my very profound gratitude to my wife for providing me with unfailing support and continuous encouragement throughout the process of classes, research, and writing of this thesis. This accomplishment would not have been possible without you. Thank you.

Luis A. Cintron

Table of Contents

Page

List of Figures

Figure Page

List of Tables

List of Algorithms

Algorithm Page

List of Acronyms

ACP	access control policy
ACR	Access Control Rule
API	application programming interface
BC	blockchain
BFT	Byzantine Fault Tolerance
BGP	Byzantine Generals Problem
BNA	Business Network Archive
BPS	Blocks Per Second
BSM	Basic Safety Message
CAN	Controller Area Network
CFT	Crash Fault Tolerance
CML	Composer Modeling Language
CSV	Comma-Separated Values
DAG	directed-acyclic graph
DB	Database
DDoS	Distributed Denial-of-Service
DL	distributed ledger
DLN	Distributed Ledger Network
DLT	Distributed Ledger Technology
DMV	Department of Motor Vehicles
DSRC	Dedicated Short Range Communication
ECU	Electronic Control Unit
GDPR	General Data Protection Regulation
GPL	GNU General Public License

GPS	Global Positioning System
HLC	Hyperledger Composer
HLF	Hyperledger Fabric
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IRC	Internet Relay Chat
ITS	Intelligent Transportation System
LLC	Logical Link Control
LTMT	Lightweight Transportation Modeling Tool
MAC	Medium Access Control
MPH	miles per hour
MSP	Membership Services Provider
MVCC	multi-version concurrency control
OBU	on-board unit
OSI	Open Systems Interconnection
OSN	Ordering Service Node
OTA	over-the-air
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
PII	Personally Identifiable Information
PKI	Public Key Infrastructure
POS	proof-of-stake
POW	proof-of-work
REST	Representational State Transfer
RFID	radio-frequency identification
RSU	road-side unit

SDK	Software Development Kit
SOAP	Simple Object Access Protocol
STSP	Smart Transportation Service Providers
SUMO	Simulation of Urban MObility
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPB	Transactions Per Block
TPF	Transaction Processor Function
TPMS	Tire Pressure Monitoring System
TPS	Transactions Per Second
UDP	User Datagram Protocol
USDOT	United States Department of Transportation
V2I	vehicle-to-infrastructure
V2V	vehicle-to-vehicle
V2X	vehicle-to-anything
VANET	Vehicular Ad-hoc Network
VM	Virtual Machine
VPKI	Vehicle PKI
VSCC	validation system chaincode
WAVE	Wireless Access in Vehicular Environments
WSM	WAVE Short Message
WSMP	WAVE Short Message Protocol

MODELING A CONSORTIUM-BASED DISTRIBUTED LEDGER NETWORK
WITH APPLICATIONS FOR INTELLIGENT TRANSPORTATION
INFRASTRUCTURE

I. Introduction

1.1 Background and Motivation

The desire for improved mobility has spawned a wide variety of devices and sensors for transportation systems. From smarter traffic light systems to traffic jam sensors, the primary goal of these devices is to decrease vehicle congestion and travel times, ensure the safety of passengers and pedestrians in transportation mediums, and increase environmental sustainability [1].

To expedite the transition to smarter Intelligent Transportation System (ITS) in major cities, organizations and institutions around the globe collaborate to set the standards for design, development, and sustainment of current and future ITSs. The United States Department of Transportation (USDOT) defines ITSs as a set of tools that facilitates a connected, integrated, and automated transportation system that is information-intensive to serve the interest of users better [1]. Proper deployment of ITSs in smart cities requires the involvement of entities other than government organizations to provide a truly integrated ecosystem.

The idea of multiple organizations involved in this extensive network of mobile and static sensors and services raises concerns due to the number of vulnerabilities potentially introduced by any of the Smart Transportation Service Providers (STSP). Also, most of these systems do not communicate with one another and are often found isolated due to lack of network services capabilities, ownership/control (company-owned and maintained), and geopolitical limitations (countries, states). Due to the centralized nature of proposed and implemented architectures, transportation stakeholders must trust each other and the data transmitted and/or received. However, networks with centralized systems and governance may face many challenges to maintain services online due to cyberattacks, network or system faults, or non-technical problems such as lack of funding by the organization in control. As a result, these technologies may benefit from Distributed Ledger Technologies (DLTs).

Emerging DLTs have changed the way transactions are conducted in environments with limited or zero trust among peers. Inherent characteristics of these technologies could prove advantageous for collecting, sharing, and storing data among ITSs by providing a decentralized collaborative platform to stakeholders. Embedding DLTs within the ITS ecosystem could fill current communication and interoperability gaps while providing governance, security, and privacy benefits not currently found in commonly used infrastructure and services.

1.2 Problem Statement

DLTs are currently being deployed for many applications. Arguably driven by the hype and popularity of cryptocurrencies, businesses and developers across the globe are pursuing solutions that rely on protocols similar or identical to those utilized by Bitcoin or Ethereum cryptocurrency networks. Unfortunately, not all use cases are a good fit for this technology. There are known limitations to blockchain (BC) or DLTs that cause them to be less efficient at processing or storing data than current mainstream alternatives, such as traditional relational databases. In particular, low operational throughput and significant latency caused by the consensus and validation algorithms often render DLTs unacceptably inefficient.

Proposals and developments for smart transportation infrastructure and services which include blockchain technology are on the rise. However, recent literature overlooks key information with respect to the acceptable use cases, the design constraints and considerations, and reference models for proper utilization of DLTs. The research described in this thesis identifies a sound and comprehensive architecture design for intelligent transportation applications utilizing a blockchain based Distributed Ledger Network (DLN) framework. It also identifies suitable applications, developmental constraints, and considerations for ITSs implementation. Drawbacks, challenges, and potential vulnerabilities that may arise as a result of this implementation are presented as well. It further demonstrates an approach for gathering and analyzing performance parameters and optimizing the overall network throughput of a given implementation.

1.3 Research Objectives

This research describes the modeling of intelligent transportation services that rely on a DLN back-end. This model is applied to two example applications: 1) vehicle records, and 2) automated accident data collection. Simulation tools are used to demonstrate how feasible the use of DLTs is to support transportation infrastructure, then test and validate security enhancements, and finally optimize network parameters that are crucial to the performance of the DLN. The research objectives of this work are outlined below:

- Understand the operation of Wireless Access in Vehicular Environments (WAVE), vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) communications for the provision of transportation-related services.

- Understand the main concepts behind distributed ledger technologies such as consensus and communication protocols, Byzantine Fault Tolerance (BFT) and

Crash Fault Tolerance (CFT), and attributes such as tampering resistance, provenance, and trust.

- Understand the motivation behind the use of DLTs in smart transportation infrastructure environments.
- Evaluate available DLT frameworks for transportation-related utility.
- Design a DLT-based high-level architecture suitable for use by ITS.
- Develop a transportation modeling environment capable of realistic interaction with a DLN.
- Identify one or more applications for the use of a DLN in ITSs that can be modeled and analyzed.
- Configure an experimental DLN that can be deployed in a transportation environment and accessed by multiple stakeholders.
- Integrate one or more ITS applications with the configured DLN.
- Evaluate the developed solution, identify issues and challenges of the application, DLN configuration or framework, and develop and test techniques for improving the performance of the network implementation.

The questions to be answered by this research in order to meet the aforementioned objectives are as follows:

- Is it feasible to enable transportation system communications utilizing DLTs?
- What applications are most suitable for utilizing a DLN in ITSs?
- What are the constraints and performance limitations of applications that rely on data from a DLN?

- Do the benefits of utilizing a DLN outweigh its constraints in terms of cost and performance?

1.4 Hypothesis

The hypothesis of this research is that it is feasible to deploy DLNs to be utilized by ITSs and provide data finality in a transportation environment. In addition, it speculates that not all applications expected in the transportation ecosystem are suitable for this technology, only those that require provenance and finality of data, suffer from a lack of trust in participants, and produce transactions at a rate that can be handled by the network endpoints and consensus algorithms.

1.5 Approach

The approach consists of standing up a consortium-based DLN infrastructure to serve as the back-end for multiple ITS applications within the same instance. This DLN allows for integration, collaboration, and maintenance of relevant data by the pre-selected parties involved in a decentralized and secure way. With the DLT framework selected, the targeted architecture defined, and the network implemented, it serves as a trusted, secure, and verifiable repository for data collected by vehicles, infrastructure, and participants. The modeled applications for the scenarios to be tested include the capability to enable the event reconstruction leading up to, during, and immediately after a vehicular accident. The data collected by sensors, either vehicles or road-side units (RSUs), comes from the source's current parameters (e.g., speed, heading, current location) and other vehicles shared via Vehicular Ad-hoc Network (VANET) V2V/V2I communications. This capability enhances vehicle forensics and is intended to improve current processes and tools for the identification of the root cause of an accident and the liable parties involved. Once the scenarios are executed under various configurations, performance data is collected and analyzed. Other findings and implications revealed during the process are documented.

1.6 Contributions

The contributions of this thesis to the field of smart transportation infrastructure and vehicles are outlined below:

- **High-level Architecture:** A 5-Tier Distributed Ledger Framework Architecture stack which defines the organization of key components for implementing the distributed network across transportation service stakeholders is presented.
- **Lightweight Transportation Modeling Tool:** This web-technology-based application provides the libraries to model transportation devices (RSUs) and vehicles, along with communications among all devices within a predefined location. It also provides a visual representation of all vehicles in motion as well as their current state and submitted transactions.
- **Hyperledger Fabric/Composer Tools:** A series of scripts were developed to ease the setup and integration of a Hyperledger Fabric network with Hyperledger Composer modeling capabilities. It allows for quick installation, maintenance, and interaction with the network and supporting applications.
- **Applications using a DLN:** It demonstrates the use of a DLN as a tool for maintaining vehicle records aimed at identifying odometer tampering fraud. Another use case demonstrated is the collection of broadcasted vehicle data over VANET that can be utilized for accident forensic analysis and provide non-repudiation of fault liability.
- **Performance Analysis:** It demonstrates an approach for measuring the performance of various blockchain network configurations under a series of loads,

identifying limitations, and performance enhancements.

- **Qualitative Analysis:** It lists design considerations, challenges, and potential vulnerabilities of the blockchain network implementation described.

1.7 Organization

This thesis is organized as follows:

Chapter II introduces the ITS and DLT concepts and terminology, then further expands on the characteristics that are of concern to this research. It defines the terminology utilized throughout this thesis as it relates to transportation infrastructure technologies and DLTs in general. It also presents ongoing related work and identifies the areas requiring further research.

Chapter III presents the idea of a transportation consortium and the use of a consortium DLT as the medium for data collaboration. Next, it describes the network architecture design and the alternatives analyzed for this implementation. It also presents the traffic simulation tool and the development decisions for generating the scenarios for testing the network. It finishes by explaining the network configurations utilizing the selected DLN platform, the data model definitions, and the distributed functions instantiated within the network.

Chapter IV lists the simulation scenario assumptions, control factors, and response variables. A description of the methodologies for conducting a network performance analysis is also presented in this chapter.

Chapter V presents the results, analysis, and observations of the experimental activities described in Chapters III and IV. It begins discussing issues found during the execution of distributed applications followed by listing the findings and performance metrics obtained through load testing the network under various configurations. The performance of the different configurations are analyzed and compared with each other to show the significance of the differences in terms of mean throughput as measured in the application layer. Possible sources of error for the findings are discussed. This chapter also lists solutions to issues and errors encountered during the stimulation of the DLN with vehicular data and discusses potential vulnerabilities of the solution at hand. Finally, it discusses benefits, drawbacks and challenges, and security and privacy concerns with the implementation presented.

Finally, Chapter VI concludes with a summary of the work presented and the contributions to the field. In addition, recommendations for those utilizing similar tools or frameworks are presented. Future work areas for this research which involve improvements to the simulation environment, upgraded components, and other performance data collection tools are also discussed.

II. Background and Related Work

2.1 Overview

This chapter provides background information and knowledge about ITSs, the technologies that enable V2V/V2I in smart transportation environments, and DLTs. It begins by describing how ITSs operate and the services they enable. Next, it highlights technologies that will enable the next generation of transportation infrastructure communication, such as WAVE. It follows with an overview on DLTs, their history, types, and key terminology for further understanding of how they operate and are able to provide capabilities such as anti-tampering, provenance, and others. A review of current literature discussing the use of DLTs in transportation infrastructure and services is presented, highlighting some of the problems in current ITS technologies and how they can be mitigated. Finally, it summarizes some of the issues with current proposed solutions and emphasizes the need for technologies that enable reliable, secure, and private communications among smart vehicles and transportation infrastructure.

2.2 Intelligent Transportation Systems

ITSs combine a collection of devices to provide services aimed at improving the quality of transportation services in a specific location or region. Three components are essential for an ITS to perform its functions; data collection, data/information transmission, and data analysis [2]. Data collection components are the sensors embedded in vehicles and RSUs such as cameras, Global Positioning System (GPS) receivers, radio-frequency identification (RFID) readers, and radars. These continuously collect data to help vehicles avoid collisions or adapt to weather conditions. Systems or sensors may share collected data with other vehicles, RSUs, or remote network services via VANET or other network communication protocols and technologies (e.g., IEEE 802.3, 3G/4G Mobile). The data received by devices is analyzed and processed to provide services such as congestion control, automatic toll collection, and collision prevention. In essence, these systems rely on information collection and dissemination to provide services to stakeholders in transportation systems. Examples of these systems are shown in Figure 1.

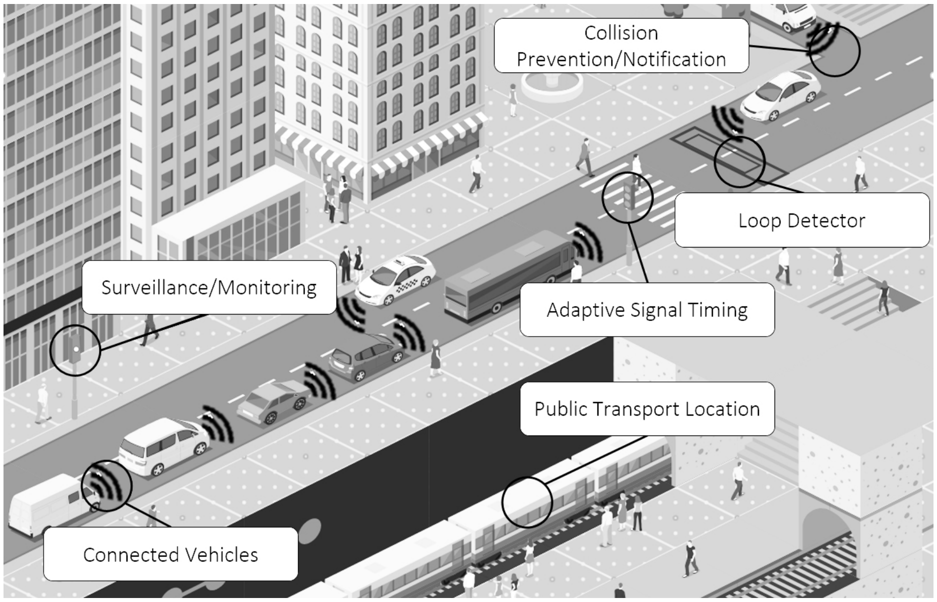


Figure 1. ITS Technologies Deployed and Integrated in Metropolitan Areas [3]

These systems allow for vehicles, pedestrians, and infrastructure components to communicate and interact with one another. Intercommunication limitations are a result of high cost of operation and maintenance, the lack of network service capabilities, issues of data ownership/control (company-owned and maintained), and geopolitical limitations (countries, states). As a result, innovative and cost-saving solutions to create a connected ecosystem of all of these systems in a more extensive network while serving as a platform for services, are currently undergoing research.

The next generation of vehicles will be equipped to conduct wireless V2V communications using the Institute of Electrical and Electronics Engineers (IEEE) 1609 Family of Standards for WAVE to communicate location, road conditions, accidents, and other information. Communication with infrastructure systems (V2I) via RSUs will also be possible. The interaction of vehicles with RSUs allows the collection of vehicle-related events that are used to provide services such as electronic tolls or accident detection and avoidance. Since RSUs typically relay received data to other networks for the services aforementioned, they rely on a separate Transmission Control Protocol (TCP)/Internet Protocol (IP) network to transmit the data collected. Similarly, vehicles can be equipped with Wide Area Network (WAN) wireless capabilities such as 4G/LTE and 5G to push notifications and event data to transportation services. These communications capabilities may allow for real-time state awareness but increase susceptible to spoofing and Sybil attacks.

2.2.1 WAVE

The IEEE 1609 Family of Standards for WAVE was developed to serve as a guideline for wireless V2V/V2I communications utilizing Dedicated Short Range Communication (DSRC) [4, 5]. The implementation of these guidelines and protocols enable what is often referred to as VANETs. These ad-hoc networks operate within the 5.850-5.925 GHz band with a 75 MHz bandwidth allocation and are enabled by both RSUs and on-board units (OBUs) installed on vehicles [6]. The OBUs are vehicle mounted devices that receive and transmit messages to and from other vehicle OBUs and RSUs.

The WAVE architecture stack is shown in Figure 2. The physical layer (PHY) is found at the bottom of the stack; it is defined by the IEEE 802.11p standard and specifies the transmission technologies and protocols. Next, the WAVE Medium Access Control (MAC) layer, also defined by the IEEE 802.11p standard, describes how the packets are transmitted across the communication channel. The Logical Link Control (LLC) layer determines which of the two protocols above it receives a packet as specified in the IEEE 1609.3 standard [7]. On top of the LLC layer sit both the WAVE Short Message Protocol (WSMP) and the IPv6 Protocol layers. The WSMP delivers received WAVE Short Messages (WSMs) to higher layers such as the WAVE message layer. The IPv6 layer delivers received packets to the TCP and User Datagram Protocol (UDP) layers on top [7]. It is crucial to note that WAVE devices can support WSMP, IPv6, or both [8].

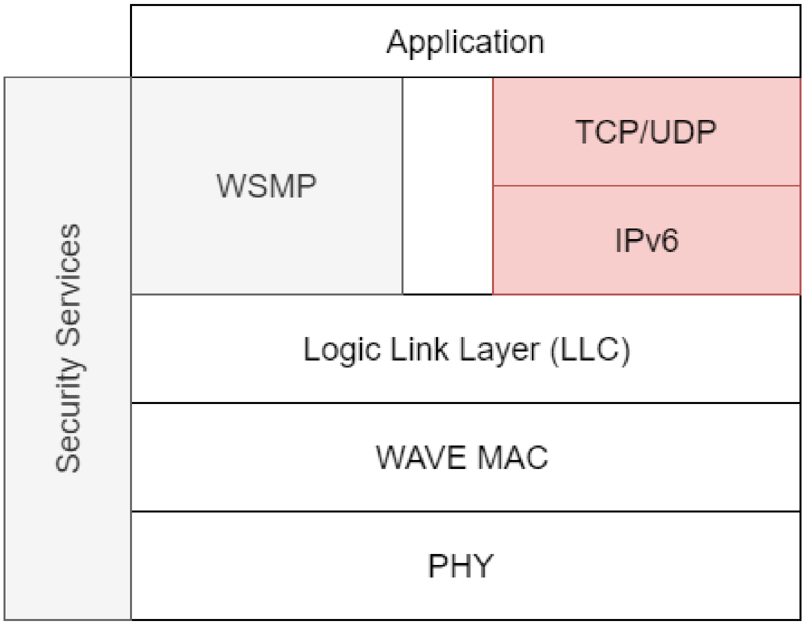


Figure 2. WAVE Protocol Stack Reference Model [5]

WAVE also defines Security Services in IEEE 1609.2 that provide cryptographic operations for secure communication among WAVE-enabled devices [5, 8]. These services support both symmetric and asymmetric encryption, hashing, and signing.

WAVE is used for the transmission of Basic Safety Messages (BSMs) which are broadcasted by devices to provide situational awareness of vehicles in an area and so that other vehicles can anticipate safety threats and avert the danger. BSMs are transmitted at a frequency of 10 messages per second with a transmission range of roughly 1 km and an average message size of 320 bytes (80 bytes for PHY + MAC + WSMP, 160 bytes Security and Certificate, 80 bytes for BSM payload)[9, 10]. Data contained in a BSM includes, but is not limited to: device ID, location (latitude, longitude), elevation, speed, heading, steering wheel angle, acceleration, and brake system status [8]. Most of these data elements are obtained from the vehicles' Electronic Control Unit (ECU) through the Controller Area Network (CAN) bus.

DSRC as defined by the IEEE 1609 Family of Standards allows for the implementation of applications that rely on the communication among vehicles and RSUs, such as: Forward Collision Warning, Emergency Electronic Brake Lights, Do Not Pass Warning, Left Turn Assist, Cooperative Adaptive Cruise Control, and Transit Signal Priority.

2.2.2 Intelligent Transportation System Applications

As the number of vehicles with vehicle-to-anything (V2X) capabilities on public roads increases, transportation infrastructure entities will lean more toward the use of infrastructure sensors and devices (i.e., RSUs) that rely on DSRC/WAVE. This new generation of transportation services and applications may include the following:

- Smart Toll Collection
- Active Traffic Management
- Vehicle Detection
- Dynamic Message Signs
- Traffic Event Dissemination
- Automated Intersection Management

- Accident Reporting and Data Collection

With the goal of building smarter transportation infrastructure and services that communicate and share data with each other, entities involved must also build the channels for communication and collaboration. However, to accomplish this task there are a number of factors that need to be taken into consideration to enable their integration. These factors include geographical and geopolitical limitations, cost, available funding, and incentives, to name a few. Because downtime of services could pose great risk to vehicle operators and pedestrians, periods of downtime due to changes in government agencies (e.g., policy, administration), funding, or technical failures must be minimized.

2.2.3 Security and Privacy in VANET and Intelligent Transportation Systems

As listed in [11], the V2X ecosystem is vulnerable to threats that affect the availability, authentication, and confidentiality of the participating devices and services. Distributed Denial-of-Service (DDoS) attacks, malware (e.g., via over-the-air (OTA) vehicle updates), and spamming are known potential threats to the availability of the transportation services that rely on V2X communications or data [11, 8]. Data spoofing, also known as a Sybil attack, (e.g., broadcast of accident data by vehicle not in area) also poses a threat. Attacks on the authentication mechanisms can include masquerading (posing as legitimate RSU or OBU) and black-holes (not relaying critical information to other OBUs and RSUs) [11]. Confidentiality within the ecosystem is vulnerable to eavesdropping and collection of broadcasted messages for data mining and location tracking thus allowing pattern-of-life modeling. Similarly, ITSs that consume OBU/RSU data are also vulnerable to DDoS attacks, malware, and spoofing. As a result, mitigating strategies such as distributed operations, containerized applications, and corroboration/validation of data can prove critical to maintain the availability and reliability of these systems without compromising the safety of its users.

Authentication mechanisms can make it difficult to employ some of these attacks through the use of Public Key Infrastructure (PKI) (supported by WAVE) where misbehaving actors are identified and their authentication certificates are revoked. Likewise, smart transportation system applications can perform post processing of data shared across multiple services to corroborate and validate the presence of vehicles and the reported data at a specified time and place. In the event of misbehavior detection, NHTSA recommends the generation of a report with the following information: reporter's identification and certificate, time, location, devices within range, speed, and suspicion type(s) [8].

Privacy is a concern in V2X communications and ITS in general. Misuse of broadcasted information by vehicles in a controlled WAVE environment in addition to detection via side-channels, cameras, and other sensors (e.g., from Tire Pressure Monitoring System (TPMS) [12]) can result in the identification of travel routines, home location, driving patterns and behaviors without a user's consent. Furthermore, this data could be mismanaged by the infrastructure operators and sold to businesses such as insurance companies triggering higher service rates without an individual's knowledge of the root cause.

2.3 Distributed Ledger Technologies

DLT refers to technologies that enable the maintenance of an append-only data structure by untrusted or partially trusted participants in a distributed fashion [13]. These are often referred to as "blockchain" technologies regardless of whether they perform actual chaining of blocks containing transactions via cryptographic means.

Some features that are inherent to DLTs are the immutability of stored data, decentralized execution, and trust of data in a trust-less environment. Figure 3 shows a high-level overview of a DLT network, where all nodes can communicate with each other and the majority of them share identical copies of the ledger. These technologies rely on a ledger in BC and/or directed-acyclic graph (DAG) data structures for guaranteeing the integrity of the data stored in the network.

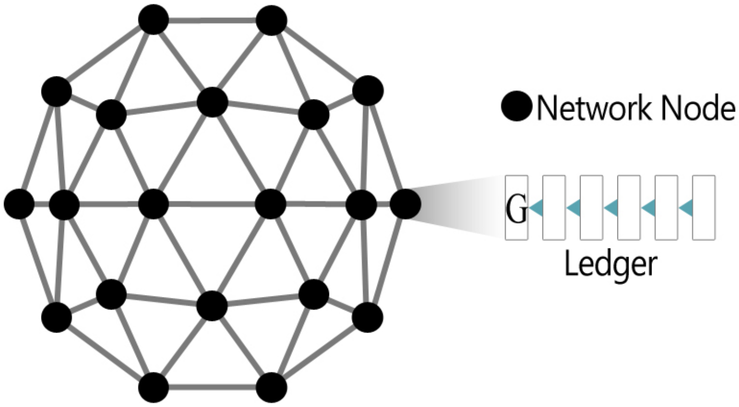


Figure 3. High-Level View of a DLN

2.3.1 Inception of Distributed Ledger Technologies

BC was proposed by an individual under the pseudonym Satoshi Nakamoto in 2008 as the underlying data structure for the now well-known cryptocurrency Bitcoin [14]. Bitcoin's network protocol allows for participants to transfer currency assets in a decentralized Peer-to-Peer (P2P) way. It relies on incentives for those nodes who take part in the creation of blocks, also known as miners. Through its use of a proof-of-work (POW) consensus algorithm, it can ensure Byzantine Fault Tolerance up to a known threshold hence solving the problem of double-spending. POW consists in miners finding a nonce that when added to a block header, the hash of the block meets a predefined condition set by the protocol in use. Through POW and validation of signed transactions, the Bitcoin network is able to create an ecosystem where participants are able to transact with one another without having to trust each other. Since then, its use has encompassed a wide range of purposes, particularly that of transfer of value (e.g., Bitcoin) and distributed computing (e.g., Ethereum) and has inspired an array of new frameworks and platforms such as those in the Linux Foundation Hyperledger Collaborative Effort.

2.3.2 Byzantine Generals Problem

"The Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others...this problem is solvable if and only if more than two-thirds of the generals are loyal..."

—The BGP by Lamport et al. [15]

Algorithms that solve the Byzantine Generals Problem (BGP) stated above under the defined conditions are said to be Byzantine Fault Tolerant (BFT). The BGP is a known issue in distributed systems and the main problem Nakamoto's consensus algorithm in Bitcoin provides a solution for. It represents a trust problem among individuals without a central authority (i.e., the generals) who need to reach agreement on an order to execute. In this specific problem, consensus is only possible if more than 2/3 of the participants are not malicious or faulty (e.g., unknown software bugs). In POW-based DLNs this is not the case due to the consensus algorithm having a self-adjusted computational difficulty that ensures consistent block creation times and prevent less than 51% of malicious computational power from creating a blockchain branch long enough to be considered as the source of truth by the rest of the network. With less than 51% of the hashing power in a network such as Bitcoin's, the malicious miners would not be able to maintain a tampered branch long enough to be accepted by the rest of the network. Although defined block times and self-adjusting difficulties for the nonce creation are considered a feature of POW algorithms, these are the main inhibiting factors for the processing of large number of transactions asynchronously, a huge disadvantage for high-throughput networks.

Other consensus solutions such as the Swirld's Hashgraph algorithm by Baird solve the BGP asynchronously through the use of a DAG [16]. Hashgraph consensus has been proven to provide true asynchronous BFT thus allowing it to process a larger number of transactions per second without wait times and high-performance computing resources. This highly efficient algorithm is reliable as long as more than 2/3 of the participating peers are honest [17].

It is important to highlight that BFT is not a strong requirement for all DLT. BFT is often a requirement when transacting with untrusted/unknown participants in a network. In networks where participants are known or the mechanisms for attribution exist, there are disincentives that keep participants from misbehaving such as: loss of employment, fines, or prosecution under the law. In such implementations, consensus can be reach through other algorithms such as indeterministic leader-selection as an alternative to BFT consensus.

2.3.3 Blockchain vs. Directed-Acyclic Graph

In the blockchain scheme, data transactions are grouped together to create a block. Each block is then appended to the ledger by a network leader selected through a consensus algorithm. The leader calculates the hash of the current block which contains the hash of the previous block in the block's header, as shown in Figure 4, and appends the block (containing the transactions) to the ledger. Once the block is added, it is considered pseudo-immutable. Immutability of data is ensured by calculating the hash for each block which contains the hash of the previous block in its header [18]. Any attempt to modify data in an earlier block would require the modification of not only that block but also all subsequent blocks for the chain to remain valid. Such an attack is an extremely difficult task in networks with many participants since the attacker must convince the majority of the participants to build on the modified fork of the chain.

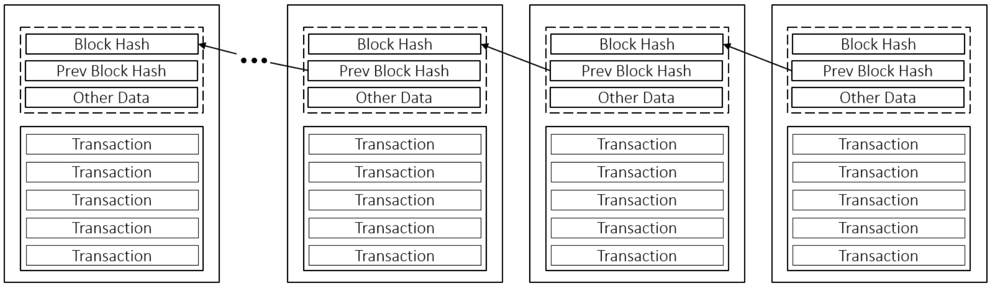


Figure 4. Blockchain DL Structure - Blocks, Block Headers, and Transaction

By contrast, a DAG is a graph data structure with no cycles that grows in one direction due to edges connecting the nodes having a direction. Therefore, in a DAG traversal, a node can only be reached once. There are various schemes and protocols for implementing distributed ledgers based on DAGs. The IOTA Foundation DAG (also known as the Tangle) does not group transactions into blocks and requires that new transactions approve two previous transactions for them to be approved and validated by other nodes in the network [19]. On the other hand, the Hedera Hashgraph DAG is "equivalent to a blockchain in which the 'chain' is continuously branching without any pruning, where no blocks are ever stale, and where each miner is allowed to mine many new blocks per second, without proof-of-work consensus and with 100% efficiency [16]". In other words, the Hashgraph maintains a graph detailing what data has been shared with what participants via gossip protocol, and the order of those transactions.

Figure 5 shows a graphical representation of a Hashgraph where each member is represented by a row of vertices and each gossip event is represented by a vertex. Once a participant receives gossip from another participant, a vertex is added to the receiving participant row with edges going to the vertex in the sender's row and the previous vertex of the receiving participant.

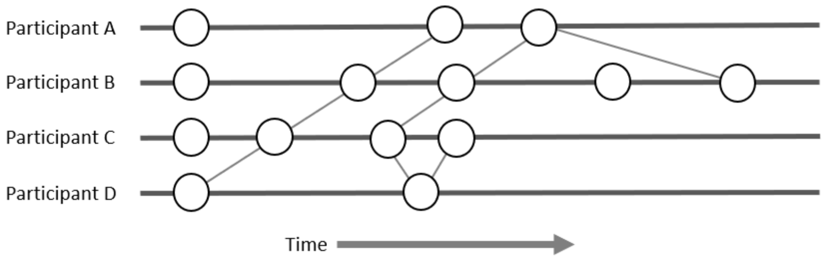


Figure 5. Hedera Hashgraph DAG DL Structure [16]

2.3.4 Categories

DLNs can be categorized based on the level of trust among the peers involved. These are categorized as Public, Private, and Consortium based depending on a series of characteristics as depicted in Figure 6. Following is a description of the specific characteristics of each of these categories.

Anyone on the Internet can access public DLNs, and their content is visible and verifiable by all of the nodes in the network [20]. These are also known as permissionless DLNs since no one individual or organization has any control about who can read the chain and submit transactions. Any participating node can elect to validate these chains and be a participant of the consensus process for deciding the next block to append to the chain and its current state [21]. Public chains operate in a low trust environment, hence requiring consensus mechanisms that minimize the potential for

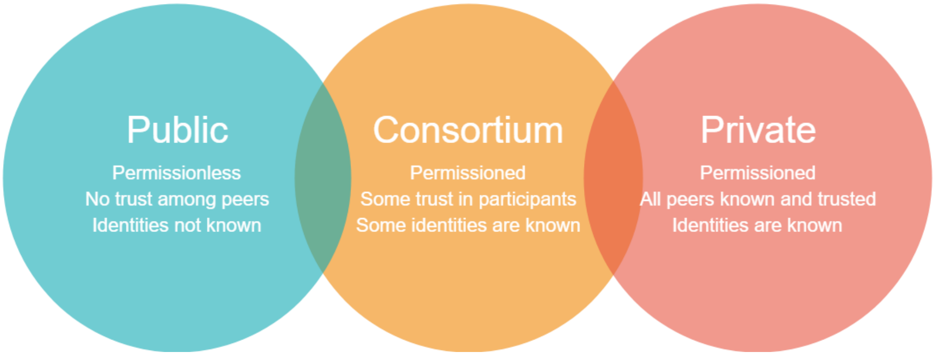


Figure 6. DLT Categories

faults, whether purposely created by colluding peers (attackers) or by software or network errors.

Conversely, private or permissioned DLNs are those where access and write permissions are maintained by a single central entity and where the network peers are highly trusted. Often, private DLs provide public readability to allow the chain to be auditable [21]. While many argue that private DLNs offer no value over the traditional data storage capabilities due to having all participants known and trusted, these can provide out-of-the-box tamper-proof logging of data and auditing capabilities [22].

Similarly, consortium chains [21], are partially decentralized solutions that constitute a hybrid between the low-trust (i.e., public blockchain) and the single highly-trusted entity model (i.e., private blockchain) [20]. The characteristics of the hybrid model allow organizations in a consortium to share transaction records without having to trust all other organizations in the network or rely on a trusted third party to facilitate the communications [23]. The consortium model fits ITS applications best for three primary reasons: its ability to make data accessible to a set of pre-selected and authenticated participants, distributed execution which allows for continuous service availability as members join or leave the network, and low-cost scalability as consortium members pay for the cost to run their nodes.

2.3.5 Consensus

Consensus—general agreement—is a critical aspect of any DLT protocol and one of the main advantages. The foundation of mutual trust among nodes in the network is based on a majority of nodes reaching consensus on the validity of the data [24]. It is required that participating nodes in a DLN reach consensus, an agreement on the next block in the chain or the order of the transactions to commit. Appropriate consensus algorithms must exhibit fault-tolerance, i.e., they must be able to perform correctly in the presence of faults that are either purposely injected by bad actors or rise as a result of system issues. In DLTs in general, reaching consensus among partially or fully untrusted nodes is a transformation of the BGP [15]. In this case, the primary goal is to ensure the ledger copies are consistent throughout the network [25] with a level of crash and/or Byzantine fault tolerance. The consensus algorithm of choice often depends on the target system and the level of trust among the members of the network. For public network implementations (i.e., permissionless DLs), algorithms such as POW and proof-of-stake (POS) allow any individual to take part in the consensus process by either solving a computationally challenging problem (i.e., POW) or staking some cryptocurrency coins (i.e., POS). In permissioned networks, consensus takes place among pre-selected nodes where the identities are known. As a result, algorithms such as Practical Byzantine Fault Tolerance (PBFT) [26], Hedera Hashgraph Consensus, and Kafka—not BFT—in Hyperledger Fabric [27] can deliver faster finality times, making the committed transactions irreversible.

2.3.6 Smart Contracts

Second generation DLTs introduced the concept of smart contracts. These are self-executing scripts that reside on the DLN and allow for automated distributed workflows [28]. These scripts can perform corroboration and validation on transactions in addition to other operations in relation to an asset or object in the ledger. As a result, they minimize the risk for trusted intermediaries between parties thus reducing the surface area for malicious activities [28]. Smart-contracts are analogous to stored-procedures in relational database systems.

2.4 Related Work

An approach for collecting VANET data for purposes of reconstructing the events taking place before and after an accident is described in [29]. The proposed solution consists of integrating vehicles with improved log recording triggering mechanisms, logging VANET communication data, and a GPS data rectification mechanism that processes data submitted by other entities. This solution maintains all logged events within the vehicle's data recorder, therefore requiring an owner's consent or a court order to get access to it. This data needs to be parsed, filtered, and appended to already acquired witness data in order to perform the forensic analysis on it. In this case, the data stored in vehicles is potentially at risk of tampering by either modification or deletion and may not always be accessible.

In [24], the authors describe a BC architecture and model for decentralized vehicle applications. Their ITS-oriented model consists of a seven-tier architecture for managing different aspects of the DLT and ITS ecosystem. On top, the application layer provides services to allow consortium members to develop applications for vehicle management, logistics, and history. Below the application layer sits the contract layer, where self-executing smart contracts can be triggered as a result of predefined conditions met by the protocol. The incentive layer follows the contract layer. This layer is where the issuance and allocation of tokens would take place. Although tokenized incentives are not a requirement in the implementation of a DLN, there are applications, such as a reputation system, where having an incentive layer would prove useful. Next, is the network layer which describes how the nodes communicate in a decentralized, P2P way, and how to validate blocks and transactions. The next tier down, the data-layer, defines the data structure used for the DL, the encryption mechanisms, hashing algorithms, and other functions such as timestamping. Lastly, the physical layer encapsulates the physical devices and entities in the ITS which include but are not limited to vehicles, roadside units, electronic toll stations, and managed lanes.

Dorri et al. [18] proposes the use of a blockchain-based DL 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 BCs such as Bitcoin, their solution clusters the network and moves the managing of the DL to nodes whose sole purpose is to broadcast and verify transactions and append blocks to the ledger. Furthermore, the author lists a series of use cases and their respective advantages, some of which include wireless software update hash checks, secure data exchange with insurance providers, and car-sharing services to name a few. Dorri et al. identified the chain of block hashes, the encryption of transactions, and the use of PKI authentication to submit transactions as significant security features of their blockchain design [18]. Also, the authors describe how the distributed architecture of the network prevents service disruption due to DDoS attacks by filtering out transactions from devices with invalid keys. Unfortunately, the research described in [18] does not include the experimentation of these services in a simulated transportation environment.

In [30], a 7-Tier blockchain architecture and framework was proposed for sharing data among vehicles on public roads. It relies on all vehicles or devices participating in the network being peer nodes. Their approach requires all peers to keep a copy of the ledger and participate in the consensus process for appending blocks to the ledger. Moreover, their approach involves a rewards layer, where a token is rewarded to vehicles “winning” consensus. Unfortunately, there are no details regarding how their proposed consensus algorithm, proof-of-driving, prevents Byzantine Faults [15] and helps with maintaining the integrity of the network. It is also unclear how their proposed approach handles a high volume of transactions made by a large number of vehicles in populated locations or how these peak times would affect their time-to-consensus. The authors also state that “the vehicle having the maximum IV-TP token, leads the communication network” leaving unclear what actions the staking leader can perform on the network (e.g., append new blocks to the ledger) or other useful purposes for the tokens rewarded.

Oham et al. in [31] described their blockchain liability attribution framework for autonomous vehicles based on a consortium of transportation and governmental organizations. This framework utilizes two partitions for communications, the operational and decision partitions, to collect or share data among different entities or sensors. Furthermore, the authors present a qualitative analysis of their framework that evaluates its resiliency against a series of malicious activities such as transaction deletion, collusion, and spoofing. A performance evaluation is presented which describes average verification and validation times for the different types of transactions. Their results show less overhead when compared to the approach described in [32]. Also, in [33] Oham et al. developed a BC based framework for auto-insurance claims and adjudication for connected and automated vehicles. The main purpose of their framework is to facilitate adjudication and claims processing. Both [31] and [33] are unclear about how the consensus algorithm operates to ensure the integrity of the network and omit design considerations for enabling such services to operate with current ITSs. Unlike the ad-hoc solutions in [31, 33, 32], the proposed implementation in this work utilizes an open-source framework that has been tested in production environments and has both community and commercial support, ongoing developmental upgrades, and can enable a number of other applications within the same platform.

2.5 Summary

This chapter presented a brief summary of the technologies involved in ITSs and how they operate. In addition, it discussed DLT categories, core concepts, and operation. It highlighted recent related work involving the use of DLTs in transportation infrastructure and vehicle communication applications. While previous research described the implementation of DLTs for enhancing ITS applications, there is no work that discusses what transportation-related applications are appropriate for implementation with DLT. There is a clear need for research describing appropriate modeling of transportation infrastructure and applications or ITS's utilizing available DLT frameworks. Similarly, the field lacks comparative evidence showing how this technology scales to allow for the millions of vehicles in public roads and other mediums when compared to current systems. Most of the research fails to show the scalability challenges with various consensus algorithms, performance metrics, or design challenges and decisions related to deploying a distributed-ledger network for ITS's to operate with. Furthermore, the proposed implementations described in this chapter lack an analysis of key factors in applying this type of technology (e.g., block size, block time, transactions per block) that affect overall performance parameters such as transaction throughput and consensus time. As a result, the effort described in this thesis focuses on the design of a DLN that enables applications that rely on data transmitted among vehicles and RSUs in the VANET ecosystem.

III. Distributed Ledger Network Design

3.1 Overview

This research investigates the use of a consortium blockchain network for transportation infrastructure, provides a detailed description of the necessary architectural layers of a sound implementation, and lists the components of the experimental network developed for the scenarios described later in this thesis. An analysis of DLT alternatives details the criteria used to select HLF as the DLT framework for implementing the transportation DLN proposed in this chapter. The network's configurable components and data model definitions are also explained.

3.2 Consortium Network for Transportation Services

As previously mentioned, the work in this thesis is based on the concept of a collaborative network of transportation-related organizations and businesses. As shown in Figure 7, these organizations could consist of government and law enforcement entities, insurance companies, vehicle manufacturers, and various other service providers sharing data within a DLN. An increasing number of ITS road sensors and the ongoing employment of V2V/V2I technologies will create a market for a new generation of interconnected services. However, in order to provide a truly integrated ecosystem for these systems, there is a need for a secure common data exchange medium where all stakeholders, from vehicle owners and operators to government agencies, can have access to, provide services, or report collected data for the purpose of improved transportation services. These services might include enhanced traffic management, accident data collection and liability analysis, and perhaps an automated toll collection system without any dedicated toll infrastructure.

Transportation entities participating in a consortium may depend on different

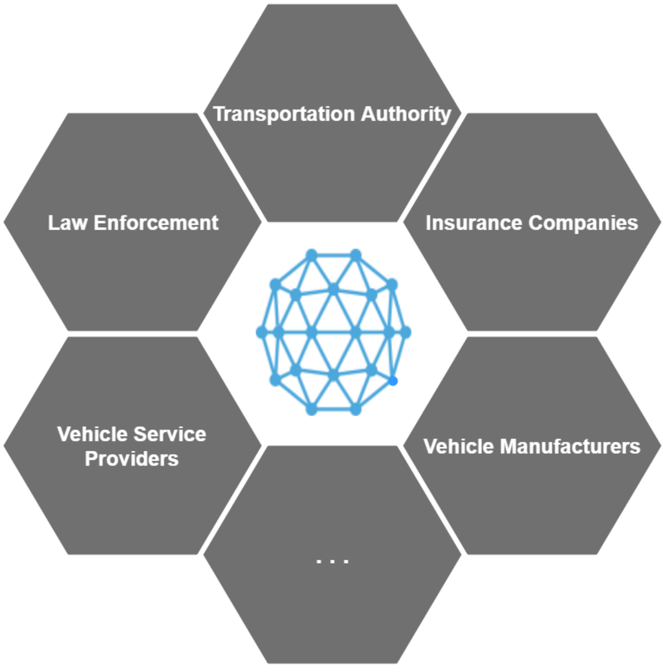


Figure 7. Transportation Infrastructure Consortium Members

types of data that can be gathered through existing sensors or previously deployed systems. They could have access to this data to provide their services, but in turn, must provide some data back that is of need or utility to other consortium members. For example, insurance companies could base an individual's insurance rate on law enforcement data. In turn, law enforcement agencies would know whether or not an individual has the coverage required by law.

The attributes of a consortium DLT provide a compelling rationale to use it as the decentralized collaborative data store among the members of a transportation consortium. This approach gives participating organizations means by which to share, validate, and corroborate data to provide applications and services for current and next generation transportation infrastructure and devices. The distributed aspect of such a network ensures a lower risk of service disruption arising from system faults or from departing members. Furthermore, it has the potential to lower the costs of infrastructure installation, maintenance, and operations since these can be amortized among entities sharing its services. Admittedly, this approach requires organizations to reach agreement on the requirements and incentives for becoming a member as well as rules and responsibilities for storing or managing different types of data such as Personally Identifiable Information (PII).

3.3 Architecture

The proposed architecture relies on a consortium (permissioned) distributed ledger framework to maintain a historical (i.e., timestamped) record of events and data captured by vehicles, RSUs, and other devices within an ITS. Data exists in the ledger in the form of transactions, signed and corroborated by vehicles, devices, and other entities in the network through smart-contracts or other applications. As a result, the grouping of transportation events stored in the DL provides substantial evidence that they indeed took place as claimed or detected.

The proposed consortium DLN architecture, illustrated in Figure 8, is analogous to the 7-layer Open Systems Interconnection (OSI) model, but tailored to a consortium DLN environment. From the bottom up, the layers of this 5-Tier stack for cooperative ITS networks are described as follows:

- The Data Layer describes the structure for storing and organizing the data such as BC or DAG, as well as the validation processes that ensure the integrity of the ledger. The technologies for storing and managing the data (e.g., Berkeley DB, CouchDB, LevelDB) are also defined in this layer. In addition to storing the ledger, database technologies can be utilized to maintain the latest state representation of the data in the ledger. This feature provides enhanced data capabilities allowing developers to query the contents of the ledger similar to relational databases without having to traverse all the blocks in the ledger.

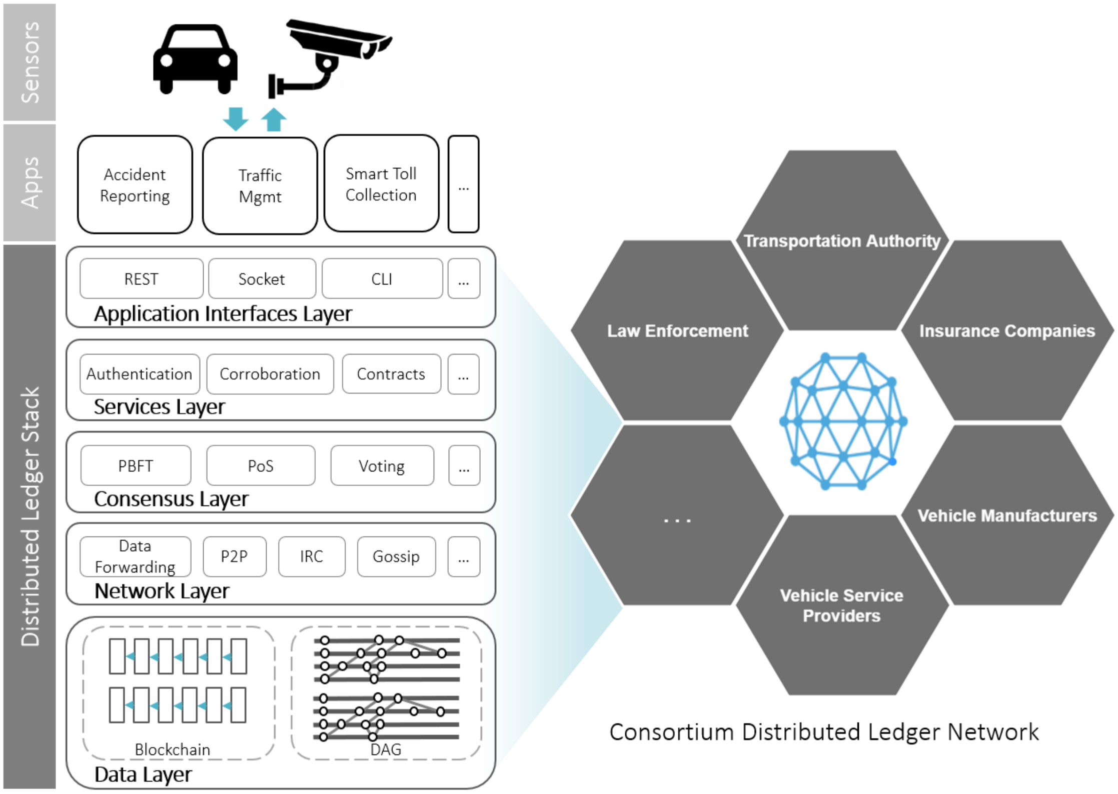


Figure 8. Consortium DLN High-Level Architecture

- The Network Layer provides the communication protocols for the distribution of transactions across the network. These can include gossip, Internet Relay Chat (IRC), and other communication protocols.

- The Consensus Layer defines the algorithms used to ensure all nodes in the network share the same ledger(s). At this layer, transactions received and endorsed are packaged into blocks which are then broadcast throughout the network using protocols from the Network Layer. Due to the partial trust within nodes in a consortium network, consensus can be performed by a smaller subset of the nodes in the network. Suitable consensus schemes for this consortium model include PBFT, POS, and Hashgraph as well as leader-selection based ones.

- The Services Layer provides the authorization mechanisms, the endorsement policies that ensure the validity of data, corroboration services that provide attestation to the events, and the smart-contract capabilities that perform self-verifying, self-executing, and self-enforcing state-response rules stored and secured at the data layer [24]. Authentication mechanisms defined in this layer can integrate capabilities as specified in IEEE 1609.2 for Vehicle PKI (VPKI) in addition to standard PKI technologies as utilized by the organizations involved.

- The Application Interfaces Layer defines the endpoints for interfacing and interacting with the network and building applications that need to read or write data from the network. This is where the endpoints for ITS applications to communicate with the transportation consortium DLN exist. Components in this layer would enable interaction with the network through network or web-based application programming interfaces (APIs) (e.g., Representational State Transfer (REST), Sockets, Simple Object Access Protocol (SOAP)) or stand-alone command-line interfaces.

This network architecture must be managed and supported by entities with the need to store or access transportation data. Entities that aim to provide services that enable and improve the ecosystem as a whole can be granted access to participate. Having access to a tamper-proof data source enables companies and organizations to provide reliable services (e.g., targeted rates from insurance providers). The quality of services depends on data that has been attested through endorsements, corroboration, and validation by other participating peers. For example, in the event of an accident, law enforcement agencies and insurance companies can create a timeline of event observations corresponding to the vehicles involved before and after the accident. This could prove advantageous at times where drivers, possibly at fault, provide an incorrect version of the events or contest being at fault.

3.4 Hardware Platform

All network nodes and simulation were executed in Virtual Machines (VMs) within a single workstation. The workstation was powered by an Intel CORE i7 vPro (7th Gen) processor (2.9GHz, 4 Cores, 8 Logical Processors) with 16GB of RAM. Each HLF node VM was allocated 2 logical processors, 2GB RAM and ran Ubuntu 16.04 64-bit operating system. The hypervisor software for hosting the VMs was VMWare Workstation 14 Pro. Docker v18.06.1 was used for the deployment of containers.

3.5 Selecting a DLT Framework

Various DLTs alternatives were analyzed in order to select an appropriate framework for the conceptual design proposed in previous sections. Table 1 lists the most appropriate DLTs considered for the implementation of the transportation network, along with the selection criteria. Due to time and development constraints, the chosen technology must provide out-of-the-box DLN application execution capabilities. It must also provide a framework for developing applications that utilize the DLN services and network API endpoints (e.g., web sockets, REST). Due to the nature of a consortium blockchain, the selected technology must provide permissioning capabilities such as user/admin membership management. Another feature it must have is the ability to define and deploy smart-contracts. These are crucial for the development of applications that execute within the DLN when specific transactions are submitted. Ideally, the selected technology would be available for use at no cost.

- **Tendermint.** Tendermint is an application and framework for creating a state-machine replication network. In essence, it provides a blockchain-based alternative for participating peers to reach consensus on the state of the application relying on this network. Its consensus algorithm is BFT hence allowing a net-

Table 1. DLT Alternatives.

Name	Executable	API/SDK Ready	Permissioned	Smart Contracts	$0 License Cost
Tendermint	✓	✓			✓
MultiChain	✓	✓	✓	✓	
Hyperledger Fabric	✓	✓	✓	✓	✓

work to operate with up to 1/3 of faulty nodes. Tendermint does not provide native permissioning or user management capabilities, however, this can be developed and integrated into existing solutions. Neither does it provide smart-contracts development. It is an open-source project with zero licensing costs for either personal or commercial deployments.

- **MultiChain.** MultiChain provides the executables and framework for the creation and deployment of private and consortium blockchain networks [34]. It has embedded identity management that enable blockchain access restriction and as of version 2.0 provides smart-contract like features known as smart-filters. Although features provided by the MultiChain framework could enable the development and deployment of a blockchain network for transportation infrastructure, version 2.0 at the time of this assessment was still an alpha version. MultiChain has a $25k/year commercial licensing cost which is required for any implementation for distribution not open sourced under GNU General Public License (GPL) v3.

- **Hyperledger Fabric.** HLF is a modular and extensible open-source framework for the deployment and operation of permissioned distributed-ledgers hosted by the Linux Foundation and maintained by IBM [35]. HLF's modular architecture allows developers to reuse available components such as membership management within their implementation. HLF relies on a scalable consensus module enabled by Apache Zookeeper and Apache Kafka, the latter one also providing publish and subscribe capabilities to the applications built within it. This framework has robust role-based user management capabilities giving developers complete control over permissions at different layers of the network. It has a framework for building smart contracts, known as Chaincode, in various languages. It provides native transaction endorsement features that are key to securing transactions in a consortium network where all participants may not trust each other. HLF has a number of support tools (e.g., Hyperledger Composer, Hyperledger Explorer, Hyperledger Caliper) that speed up the setup, development, analysis, and deployment of applications on top of it. There are no costs to deploy HLF for personal or commercial purposes on your own infrastructure.

Based on the criteria described above, HLF was the technology selected for this experiment. In addition to the features previously described, it allows for multiple channels ("blockchains") within the same platform network for different applications and has private data capabilities. The framework provides a Software Development Kit (SDK) for popular languages such as Go, Java, and JavaScript (NodeJS). Its modular architecture makes it an attractive framework for entities not interested in re-developing their technologies to deploy a DLN for their services. In early 2019, HLF v1.4 Long Term Support was released with promising new capabilities and improved features, but was not adopted for this effort.

3.6 Frameworks and Tools

The following frameworks and tools were utilized for creating the blockchain network, defining the data-models and smart contracts, and querying the ledger for data.

3.6.1 Hyperledger Project Technologies

The Hyperledger Project is a collaborative effort hosted by The Linux Foundation that aims to develop and support open-source enterprise DLTs [36]. Currently it hosts over 10 projects which include frameworks and tools for the design, development, deployment, and testing of DLNs and distributed applications. The next sub-sections described the Hyperledger Project framework and tools utilized for this experimental network.

3.6.1.1 Hyperledger Fabric

HLF (v1.2) was chosen primarily because its modularity allows developers and architects to tailor its different layers, such as methods for validation, consensus, and the distributed-ledger data structure, to meet an organization's needs [35]. Furthermore, HLF allows the creation of a consortium-based network of peers where organizations can manage their own users' permissions.

Key terminology related to HLF is defined below:

- **Consensus:** HLF supports pluggable consensus and it can support different types per channel. Consensus in HLF takes place among the Ordering Service Nodes (OSNs), commonly referred to as orderers, and is performed by selecting a leader among them 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. The framework deployed in this research uses Apache Kafka and Apache Zookeeper for consensus since it is the only consensus module considered ready for production environments. The Zookeeper-Kafka consensus at its core enforces the ordering of transactions in a block. Blocks are created when a predetermined number of transactions are received or when a timer since receiving the first transaction for a block has expired (block timeout). Once one of these two actions happen (whichever one happens first), a block is delivered to the peer nodes [37].

- **Channel:** A channel in HLF is a private blockchain where only the specific channel participants have access to and can interact with it [38]. Participation is managed via authentication and Access Control Rules (ACRs).

- **Chaincode:** Also known as smart-contracts, chaincode is the code/service invoked by an application interacting with the HLF network that manages access and modifications to the ledger. It is installed on peer nodes to work on one or more available channels [38].

- **Endorsement:** In HLF, endorsement refers to peer nodes simulating the execution of a chaincode transaction and communicating the response back to the originator, along with their signature to provide proof of valid execution result [38]. Endorsement policies define a transaction endorsement requirement in the form of Boolean expressions over participating organizations [37].

- **Membership Services Provider (MSP):** The MSP provides cryptographic (PKI-based) credentials to the HLF participants for authentication and transaction processing [38].

- **Peer Node:** A network node that executes the chaincode and maintains a copy of the ledger. Peers can be identified as endorsers in order to participate in the endorsement of a transaction and can also be identified as anchors which allow them to be discovered by and communicate with all other peers.

- **Orderer Node:** OSNs or Orderers are nodes that participate in the consensus phase. They receive transactions from other peers, create blocks and broadcast them to all other endorsing and committing peers [37, 38].

- **Access Control Rules:** ACRs can be deployed to control which participants have access to which assets and the conditions under which that access is granted or denied. This allows certain entities to perform actions on assets they own, or update the state of an asset given a set of rules. Although a feature available through Hyperledger Composer, the modeled network does not employ the use of ACRs, all vehicles and RSUs are assumed to have the add/update rights to the events they are involved in.

HLF networks follow the execute-order-validate-commit model for transactions shown in Figure 9. Transactions are received in the *execute* phase where they are passed through chaincode, if needed, while at the same time the receiving peer sends a request for endorsement. Requests for endorsement are sent only if such requirement has been set when instantiating chaincode. In order for the transaction to be valid, the endorsement execution at a different peer node must yield the same output as the same execution at the originating node. Once the transaction has received all endorsements, it is sent to an orderer for consensus, entering the *ordering* phase. At this stage the transactions are ordered by their timestamps and await inclusion in a block. A block is created whenever the first of the following two conditions is met: the block is filled with the maximum number of transactions allowed or a block timer expires, i.e., block timeout. Once the block is cut, it is broadcast to other nodes for validation. During the *validation* phase, peer nodes ensure that the received transaction has valid endorsement signatures and meets the pre-defined endorsement policy requirement through a process known as validation system chaincode (VSCC). Another type of validation that takes place during this stage is multi-version concurrency control (MVCC) [39] validation, which checks if the state of the object being modified by a transaction in a block has not changed since the transaction was executed and tags it valid or invalid. Once validation has completed, the transaction is *committed* to the ledger and the state database is updated to reflect the creation or modification of the data asset in question. Finally, any events specified in the blocked and committed transactions are then emitted.

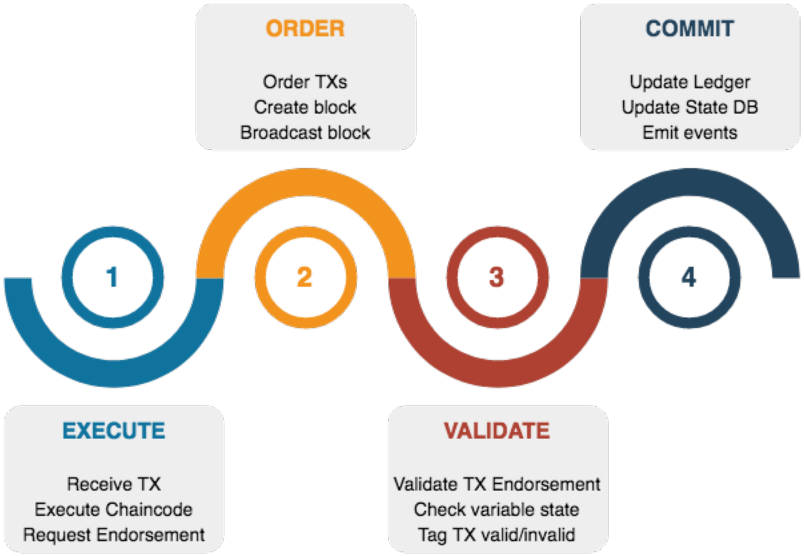


Figure 9. Hyperledger Fabric Transaction Workflow

3.6.1.2 Hyperledger Composer

Hyperledger Composer (HLC) (v0.20.3) is a toolset and framework aimed at facilitating the development and execution of blockchain networks and services [40]. It was utilized to model and deploy the ITS chaincode and network services of this setup. It includes a “Composer-Playground” tool to view and interact with world-state data and perform upgrades to the services. Composer has the capability to stand up a REST server that interfaces with the HLF network to provide a web API.

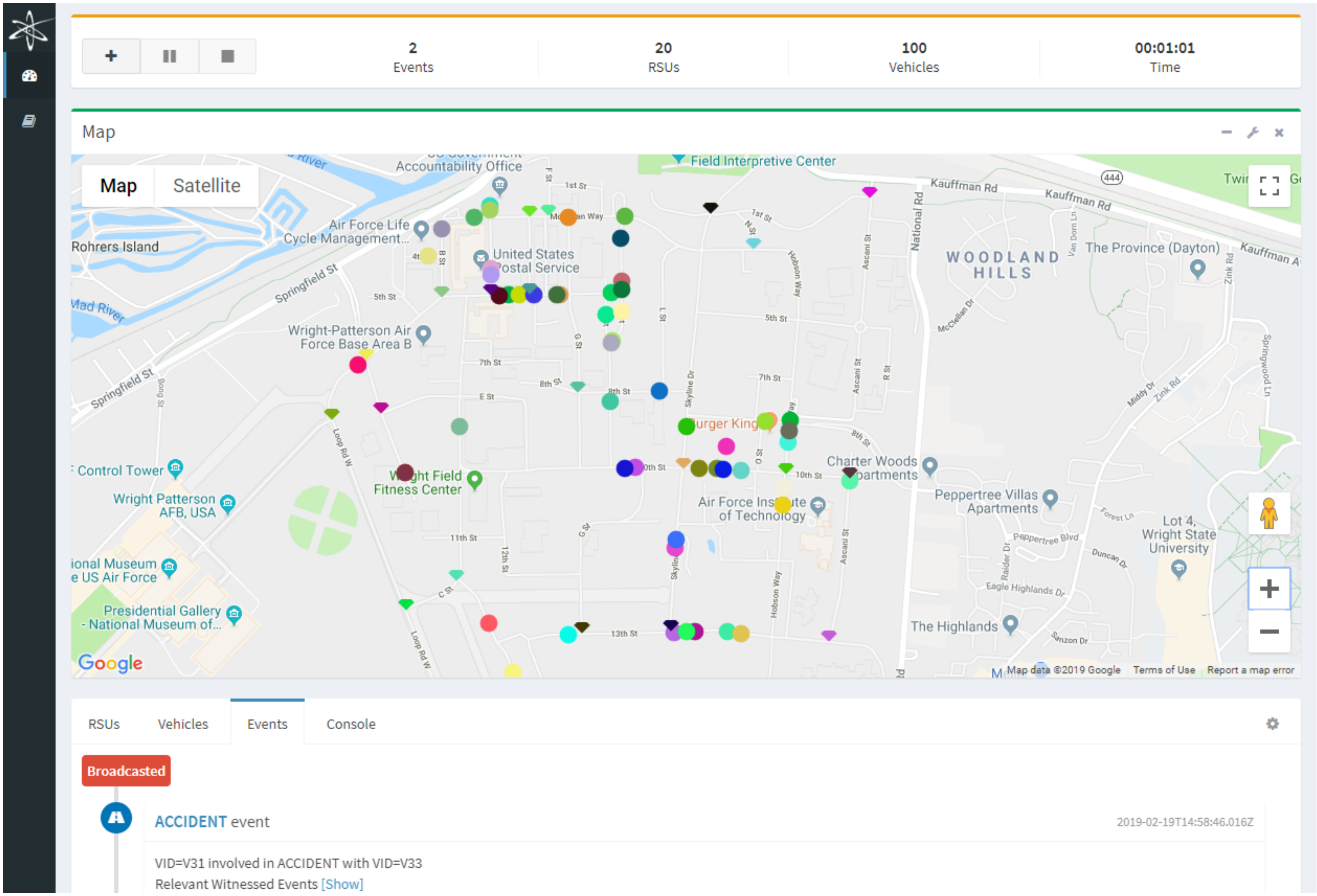
3.6.1.3 Hyperledger Explorer

The Hyperledger Explorer (v3.5) project provides a web application to interact and query underlying data from the blockchain that is not available through the HLC generated services [41]. Block information such as hashes, timestamps, and transactions are made available via its services layer which can be used for computing performance data such as network throughput in Transactions Per Second (TPS).

3.6.2 Lightweight Transportation Modeling Tool (LTMT)

This tool, shown in Figure 10, was developed for the work described in this thesis in order to provide a simple environment for the modeling and simulation of moving vehicles and their communication with infrastructure and other vehicles. Popular alternatives such as the Simulation of Urban MObility (SUMO) [42] allow the generation of vehicle scenarios at a much larger scale but do not have native capabilities to interact with web APIs.

Lightweight Transportation Modeling Tool (LTMT) relies on Google Maps to display vehicles and infrastructure sensors (e.g., RSUs) in a map and the Google Directions API to generate vehicles routes around a pre-defined region. It is built on JavaScript and NodeJS and can be deployed as a web or stand-alone (requires Internet access) application. In this experiment, the LTMT generates the vehicle traffic and accident scenarios and acts as an application client that interacts with the DLN. Its main purpose is to simulate vehicle and infrastructure behavior and communications during the event of an accident.


+
2
Events
20
RSUs
100
Vehicles
00:01:01
Time
Map
Map
Satellite
Accountability Office
Field Interpretive Center
Kauffman Rd
Air Force Life Cycle Management...
Rohrers Island
United States Postal Service
WOODLAND HILLS
The Province (Dayton)
Springfield St
Mad River
Wright-Patterson Air Force Base Area B
5th St
7th St
8th St
Burger King
Control Tower
Wright Patterson AFB, USA
Wright Field Fitness Center
Charter Woods Apartments
Air Force Institute of Technology
Peppertree Villas Apartments
Lot 4, Wright State University
ional Museum e US Air Force
Presidential Gallery - National Museum of...
The Highlands
13th St
Google
Map data ©2019 Google
Terms of Use
Report a map error
RSUs
Vehicles
Events
Console
Broadcasted

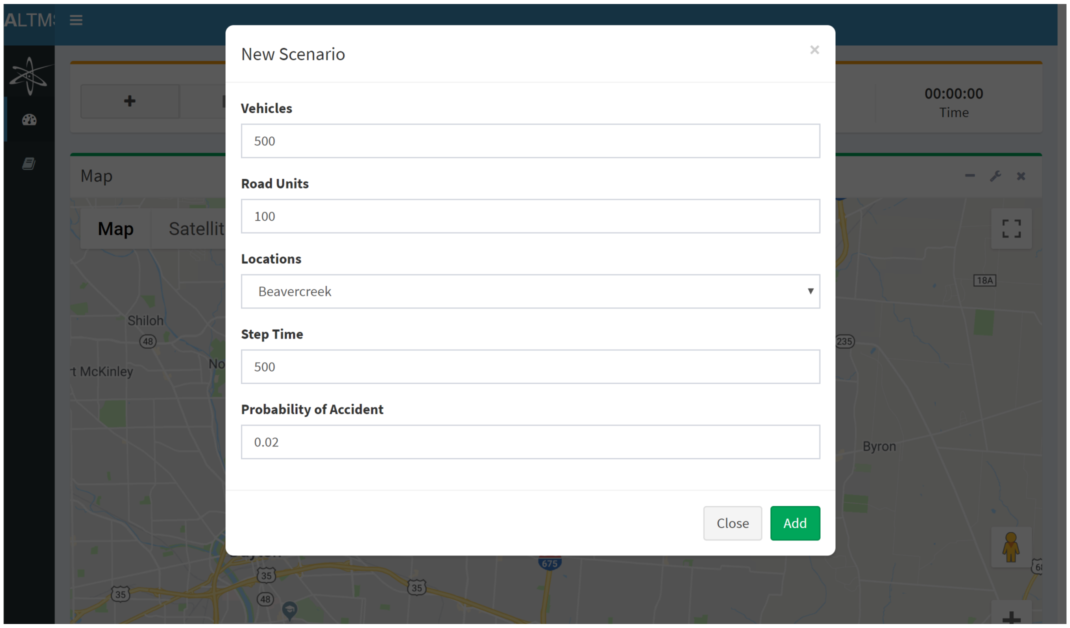
ACCIDENT event

2019-02-19T14:58:46.016Z

VID=V31 involved in ACCIDENT with VID=V33

Relevant Witnessed Events [Show]

Figure 10. LTMT User Interface (Map data: Google)



New Scenario

Vehicles: 500

Road Units: 100

Locations: Beavercreek

Step Time: 500

Probability of Accident: 0.02

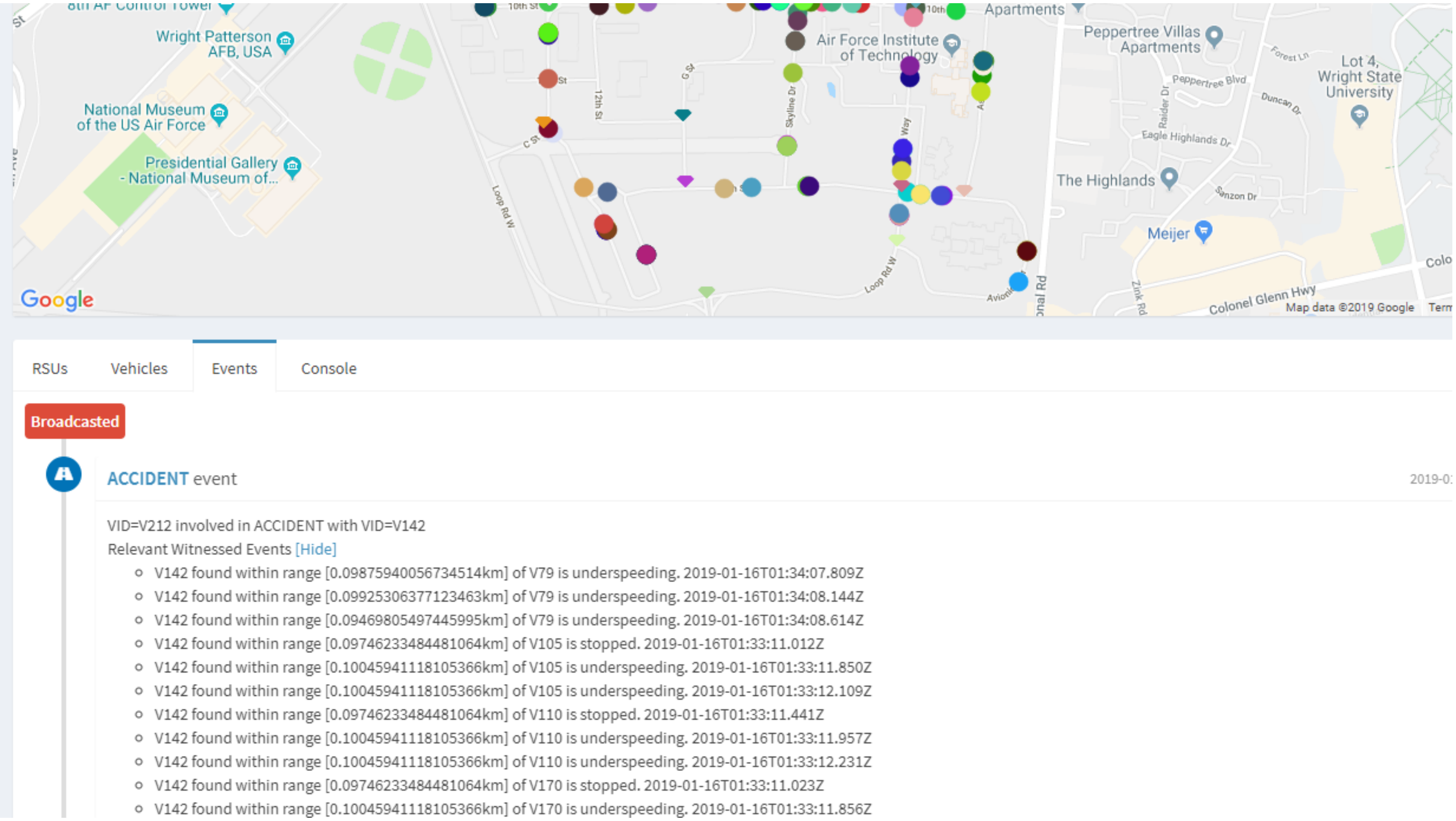
Figure 11. LTMT Scenario Generation

The tool relies on a list of addresses or coordinates provided by the users from which the vehicle paths are generated. Users can manually add vehicles to a scenario by specifying the source and destination locations or they can be automatically added with origin and destination addresses selected from the pool of addresses for the pre-determined location with the scenario generation wizard (Figure 11). At scenario initialization, it queries Google's Direction API for the path of travel and road conditions that affect the speed and route of the vehicle. Vehicles are uniquely identifiable and their location, current speed, and heading parameters are calculated and updated in every timestep and broadcast in a BSM, (shown in Table 2). The BSM parameter choices are based in the discussion of V2V/V2I BSM data in [29] and in accordance to [9, 10]. Similarly, RSUs can be manually added by specifying the placement location or automatically during scenario generation. However, RSUs do not broadcast BSMs, instead, they collect data to perform a service defined by the developer.

Table 2. LTMT Vehicle BSM Content.

Parameters
ID
Date - Time
Location: Latitude, Longitude (Decimal Degrees)
Speed (miles per hour (MPH))
Heading (Degrees)

As vehicles are en-route to their destination, they broadcast and receive BSMs from vehicles within a distance of 100 meters and collect speed, heading, and ID information from broadcast messages. BSMs are broadcast/received during every time step. This tool does not employ expiring pseudonyms as means to identify vehicles, and all IDs are static for the duration of the scenario. RSUs can also broadcast or receive vehicles nearby and categorize them as misbehaving based on the information they broadcast and what the detecting sensor sees. Misbehavior occurs when a vehicle is detected speeding, underspeeding, or stopped. These misbehavior activities are defined in a collection of rules. These rules are evaluated during every vehicle detection and communication step. Both vehicles and RSUs keep a log of received messages and detected misbehaving sensors containing the location of detection, the detected vehicles broadcast data (ID, date-time, location, speed, heading), the distance of the vehicle from the source sensor, the type of misbehavior detected, and a list of other sensors within range. Figure 12 shows an accident event as broadcast in the tool with collected misbehavior witnessed data.


8th AF Control Tower
Wright Patterson AFB, USA
National Museum of the US Air Force
Presidential Gallery - National Museum of...
Air Force Institute of Technology
Apartments
Peppertree Villas Apartments
Lot 4, Wright State University
The Highlands
Meijer
Colonel Glenn Hwy
Google
Map data ©2019 Google
RSUs
Vehicles
Events
Console
Broadcasted

ACCIDENT event 2019-0:

VID=V212 involved in ACCIDENT with VID=V142

Relevant Witnessed Events [Hide]

- V142 found within range [0.09875940056734514km] of V79 is underspeeding. 2019-01-16T01:34:07.809Z
- V142 found within range [0.09925306377123463km] of V79 is underspeeding. 2019-01-16T01:34:08.144Z
- V142 found within range [0.09469805497445995km] of V79 is underspeeding. 2019-01-16T01:34:08.614Z
- V142 found within range [0.09746233484481064km] of V105 is stopped. 2019-01-16T01:33:11.012Z
- V142 found within range [0.10045941118105366km] of V105 is underspeeding. 2019-01-16T01:33:11.850Z
- V142 found within range [0.10045941118105366km] of V105 is underspeeding. 2019-01-16T01:33:12.109Z
- V142 found within range [0.09746233484481064km] of V110 is stopped. 2019-01-16T01:33:11.441Z
- V142 found within range [0.10045941118105366km] of V110 is underspeeding. 2019-01-16T01:33:11.957Z
- V142 found within range [0.10045941118105366km] of V110 is underspeeding. 2019-01-16T01:33:12.231Z
- V142 found within range [0.09746233484481064km] of V170 is stopped. 2019-01-16T01:33:11.023Z
- V142 found within range [0.10045941118105366km] of V170 is underspeeding. 2019-01-16T01:33:11.856Z

Figure 12. LTMT Accident Event Notification and Misbehavior Data Collection (Map data: Google)

In its current state, the tool can trigger accidents between two vehicles within collision distances based on either a defined probability-of-accident value or violation of rules that define misbehavior activity such as speeding, underspeeding, or stopped. Vehicles are considered speeding when traveling at a speed of 10 MPH over the defined speed limit and underspeeding when traveling at a speed of 10 MPH under the speed limit. Developers can modify LTMT's libraries to redefine traffic and misbehavior rules, or to communicate events to other applications via HTTP REST APIs or WebSockets.

The LTMT was limited to a maximum number of 600 vehicles and RSUs that can be added to a single scenario when generated through the browser interface. This limitation is due to JavaScript's single-thread allocation per browser instance and introduced latencies in the browser's JavaScript engine event-loop which results in delayed vehicle movements, delayed communications, and an overall poor user-experience. However, scenarios that require more sensors can be generated by running the same scripts without the user-interface bindings using NodeJS and exporting the results or events per timestep to a file. This file can be consumed by other applications for further processing or analysis.

3.6.3 Network Architecture based on Hyperledger Technologies

Figure 13 below shows how the selected DLTs implement the different layers of the proposed architecture. It also depicts the layers abstracted by HLC to provide a developer friendly environment, and the additional capabilities it provides beyond HLF. Hyperledger Explorer, on the other hand, does not utilize HLC's abstraction layer but instead communicates with HLF's services layer directly to query information about blocks and transactions stored in the blockchain.

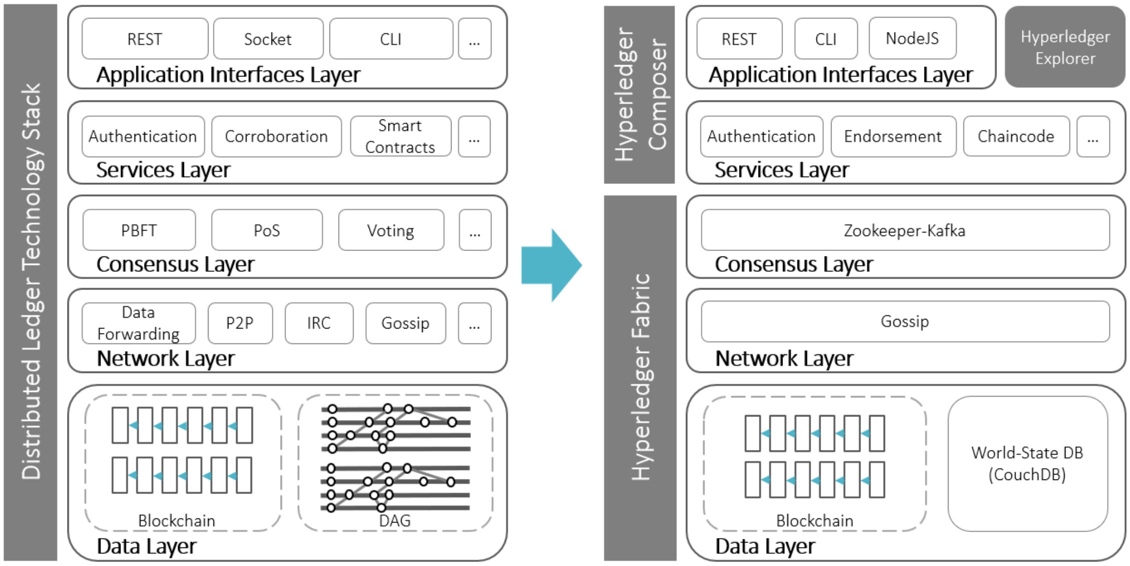


Figure 13. Hyperledger Fabric/Composer Based Network Architecture

3.7 Network Design

The baseline configuration for this HLF network with a Zookeeper-Kafka cluster is described in Table 3. Figure 14 shows the network topology of this configuration. For purposes of this experiment, MSP, OSN, and peer containers for each organization are initiated in the same VM. In addition, each organization's VM executes Composer REST Server to expose the web API. All VMs were configured with static IP addresses and the Docker container configurations were set to utilize host network mode. In a production HLF network, each consortium organization will have at least the components described. Most likely, these will employ multiple peer nodes within their bounds to distribute the usage of the services.

Table 3. Baseline HLF Network Configuration.

Parameters	Values
Participating Orgs	3
Orderer Nodes	3
Peer Nodes	3
Channels	1
Zookeeper-Kafka Cluster (Consensus)	3 Zookeeper Nodes 4 Kafka Nodes
World-State Database (DB)	CouchDB
Block Size	99MB OR 10 Transactions Per Block (TPB)
Block Timeout	2 seconds
Endorsement	ORG1 & ORG2 & ORG3

The network was initiated with three organizations (consortium members) since that is the minimum number required for creating a partial-trust environment where a blockchain or DLN would be most beneficial (not enforced by HLF) and due to limited computing resources available. There are more efficient ways to store and manage data for a single organization than utilizing DLTs, for example, distributed databases. When two organizations collaborate with each other, this collaboration assumes full trust between the two entities and there is no need for endorsement or consensus in data distribution. In a three-organization consortium, there is the possibility that not all organizations trust each other, therefore bringing value to endorsement and a leader or voting based consensus.

The number of nodes for the Zookeeper-Kafka cluster configuration is the minimum number of nodes required to setup Zookeeper-Kafka consensus in HLF v1.2. For Kafka nodes, four is the minimum number of nodes to exhibit CFT [43]. For Zookeeper nodes, the number must be odd to "avoid split-brain scenarios and larger than 1 in order to avoid single point of failures" [43]. As a result, three Zookeeper nodes were configured. As shown in Figure 14, the Zookeeper-Kafka cluster, com-

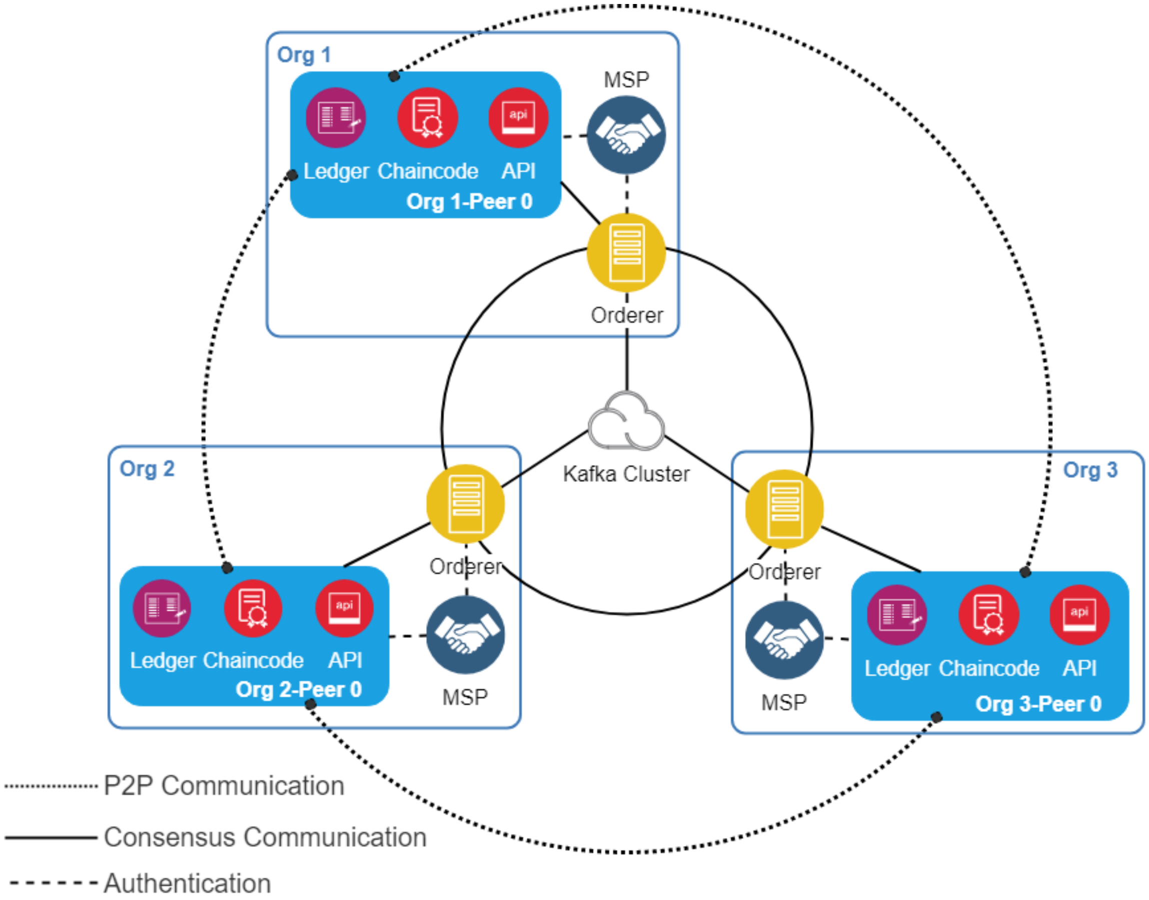


Figure 14. Experimental Hyperledger Fabric Network Configuration

posed of multiple Zookeeper and Kafka nodes that need not be co-located, can be considered a pitfall due to the centralization. It begs the question of who is in charge of the maintenance and costs to operate this cluster. The consortium as a whole could split the costs of maintaining these crucial components of the network.

3.7.1 Hyperledger Fabric Network Initiation

Once the HLF framework has been downloaded and installed in the VMs for organizations 1, 2, and 3 respectively, the cryptographic materials (certificates, genesis block, channel block) using the configuration files included in Appendix A and

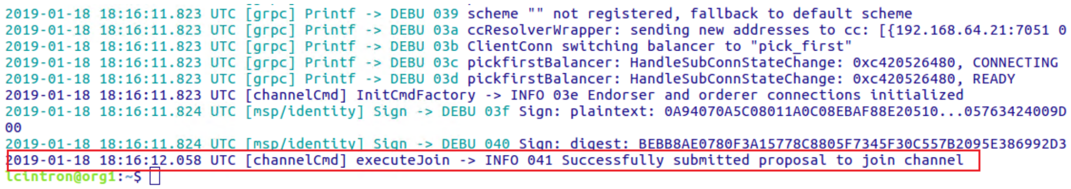
Appendix B are generated. Next, the network is brought up in the following order (specific commands are enumerated in Table 4):

1. Start Zookeeper-Kafka VM and cluster containers
2. Start Org1 VM. Start Org 1 HLF Orderer, MSP, CouchDB, and Peer containers. Create and join channel.
3. Start Orgs 2 and 3 VMs. Start Orgs 2 and 3 respective HLF Orderer, MSP, CouchDB, and Peer containers. Fetch channel configuration and join channel.

Table 4. Commands to Start HLF.

VM Name	Commands
Zookeeper-Kafka	\$docker-compose -f $< DOCKERFILE >$ up
Org1	\$docker-compose -f $< DOCKERFILE >$ up; \$peer0.org1.afit.edu peer channel create -o orderer1.afit.edu:7050 -c composerchannel -f /etc/hyperledger/configtx/composer-channel.tx; \$peer0.org1.afit.edu peer channel join -b $< BLOCKNAME >$
Org2 and Org3	\$docker-compose -f $< DOCKERFILE >$ up; \$peer$x$.org$x$.afit.edu peer channel fetch config -o orderer$x$.afit.edu:7050 -c composerchannel; \$peer0.org1.afit.edu peer channel join -b $< BLOCKNAME >$

The commands in Table 4 are in a script located in each VM, and executed after booting up. The Docker configuration file utilized for an organization is shown in Appendix C and Appendix D shows a version of the script for starting a node VM based on the Docker configuration. Peers that successfully joined the network displayed the message shown in Figure 15. The network is now ready for the deployment of the Hyperledger Composer data models, chaincode, and REST server.



```
2019-01-18 18:16:11.823 UTC [grpc] Printf -> DEBU 039 scheme "" not registered, fallback to default scheme
2019-01-18 18:16:11.823 UTC [grpc] Printf -> DEBU 03a ccResolverWrapper: sending new addresses to cc: [{192.168.64.21:7051 0
2019-01-18 18:16:11.823 UTC [grpc] Printf -> DEBU 03b ClientConn switching balancer to "pick_first"
2019-01-18 18:16:11.823 UTC [grpc] Printf -> DEBU 03c pickfirstBalancer: HandleSubConnStateChange: 0xc420526480, CONNECTING
2019-01-18 18:16:11.823 UTC [grpc] Printf -> DEBU 03d pickfirstBalancer: HandleSubConnStateChange: 0xc420526480, READY
2019-01-18 18:16:11.823 UTC [channelCmd] InitCmdFactory -> INFO 03e Endorser and orderer connections initialized
2019-01-18 18:16:11.824 UTC [msp/identity] Sign -> DEBU 03f Sign: plaintext: 0A94070A5C08011A0C08EBAF88E20510...05763424009D
00
2019-01-18 18:16:11.824 UTC [msp/identity] Sign -> DEBU 040 Sign: digest: BEBB8AE0780F3A15778C8805F7345F30C557B2095E386992D3
2019-01-18 18:16:12.058 UTC [channelCmd] executeJoin -> INFO 041 Successfully submitted proposal to join channel
lcintron@org1:~$ 
```

Figure 15. Hyperledger Fabric Peer Node Successfully Joins Channel

3.8 Transportation Application Design

Data models for storing data relevant to the consortium entities and distributed chaincode (functions) are defined and deployed using HLC. Data model definitions and chaincode are bundled into a Hyperledger Composer Business Network Archive (BNA). The BNA file also contains an access-control rules list for the applications and query definitions that are exposed through the Composer generated APIs. The deployment of this file creates a new container on the peer node that utilizes the HLF network to store the data and execute the chaincode in the specified channel.

3.8.1 Data Models

HLC has its own object-oriented modeling language, Composer Modeling Language (CML), which was used to define the data structures required for deployment of the applications to the network. The main data models and respective attributes defined in CML for the applications described in this experiment are shown in Table 5; Appendix E contains all the data model definitions deployed to the network.

3.8.2 Chaincode

Chaincode in HLF can be developed in Java, Go, and JavaScript. In this implementation, chaincode is developed and executed through a layer of abstraction provided by HLC using ES 5 JavaScript. API classes and functions provided by HLC

Table 5. Main Data Models and Attributes.

Name	Type	Attributes
WitnessedVehicleData	concept	+observedVehicle:*Vehicle* +source:*Sensor* +location:*Location* +eventTimestamp:*DateTime* +eventId:*String* +speed:*Double* +heading:*Double* +distanceFromSource:*Double* +behavior:*String[]*
Sensor	asset	+id:*String* +type:*SensorType*
Vehicle *extends* Sensor	asset	+odometer:*Double* +eventsInvolved:*RoadEvent[]*
RSU *extends* Sensor	asset	+id:*String* +location:*Location*
RoadEvent	asset	+id:*String* +location:*Location* +eventTimestamp :*DateTime* +type:*EventType* +vehiclesInvolved:*Vehicle[]* +witnessedData:*WitnessedVehicleData[]* +validatingReports:*String[]* +seenVehicleReporters:*String[]* +uncorroboratedReports:*String[]* +sourceSensor:*Sensor*

for chaincode development are not the same as those in the HLF NodeJS SDK. However, HLF native function calls can be made from the HLC layer. Chaincode logic in HLC is done through Transaction Processor Functions (TPFs) that are bundled within the BNA that is deployed to the HLF network to provide the transportation related capabilities and services. TPFs are automatically invoked when transactions are submitted utilizing the APIs generated by HLC. In the experimental network implementation, the deployed chaincode references the data models in Table 5 and describes how to use the transaction objects to update vehicle information such as last known odometer readings in addition to creating road event reports (e.g., accident) and appending witness data to those reports. The procedures defined in chaincode must be passed arguments (if required) that are objects of transaction type classes. They can be used to verify the existence of the sensors submitting the transaction, perform data validation before creating the asset (e.g., bounds validation), and decide when to broadcast an event.

3.8.3 Endorsement Policy

Endorsement policies are passed as a parameter to the command that starts the HLF network layer defined in the BNA or defined in the HLF when instantiating chaincode. Calling the

$ composer network start

command without specifying the endorsement policy results in standing up a network where all transactions require endorsement by all peers identified as endorsing peers in the connection profile (See Appendix H for example).

The performance of the network, loaded with a series of scenarios, will be measured with the default endorsement policy and a *2-Of* endorsement policy. The 2-Of endorsement policy shown in Figure 16 requires a TPF with given transaction to be executed and signed by at least two different organization members of the network. This will help determine whether or not the changes to endorsement policy through Hyperledger Composer result in statistically significant improvements.

In a three-organization consortium network, a 2-Of policy is the least restrictive endorsement policy without completely removing endorsement. Removing the requirement for transaction endorsement in these types of networks means that all participating organizations fully trust all other organizations execution, however, these can be attempting to execute a transaction on outdated or corrupted blockchain data and their execution result may not be the same as the result on other peer nodes. Endorsement in HLF provides a mechanism to ensure deterministic and verifiable results in the execution of transactions among a subset of participating nodes in the network.

```
{
        {"role": {"name": "member","mspId": "Org1MSP"}},
        {"role": {"name": "member","mspId": "Org2MSP"}},
        {"role": {"name": "member","mspId": "Org3MSP"}},
    ],
    "policy": {
        "2-of": [{"signed-by": 0},{"signed-by": 1},{"signed-by
           ": 2}]
    }
}
```

Figure 16. 2-Of Endorsement Policy Definition

3.8.4 Access Control List

The access control list file specifies the access rules for the different network participants defined as part of the data models or known to the HLF network (e.g., network admins). These define which participants have access to perform a specific operation to a resource under a specified condition. For example, the ACR depicted in Figure 17 gives vehicle owners read access to all transactions submitted by their vehicles.

```
rule historianAccess {
    description: "Only allow vehicle owners to read historian
       records referencing transactions they submitted."
    participant(p): "org.afit.transportation.VehicleOwner"
    operation: READ
    resource(r): "org.hyperledger.composer.system.
       HistorianRecord"
    action: ALLOW
}
```

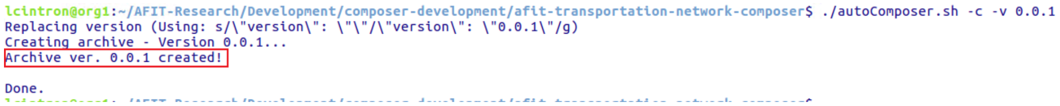
Figure 17. Access Control Rule Example

For purposes of this experiment, ACRs will not be defined and all sensors and users are assumed to meet the appropriate access requirements to read or create transactions.

3.8.5 Hyperledger Composer Archive Deployment

In order to deploy the chaincode and data models defined using the HLC tools, the HLF network must be up and running. Deploying the BNA to the HLF network requires the execution of a series of commands which is facilitated by the *autoComposer.sh* script (Script shown in Appendix G). The *autoComposer.sh* commands must be executed from the directory containing the BNA files in the following order:

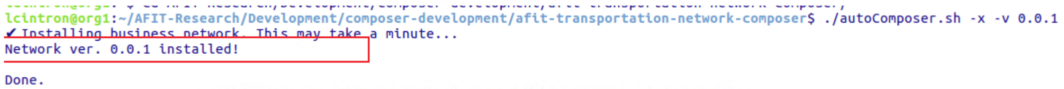
1. Generate the BNA file (Figure 18):



```
lcintron@org1:~/AFIT-Research/Development/composer-development/afit-transportation-network-composer$ ./autoComposer.sh -c -v 0.0.1
Replacing version (Using: s/\"version\": \"\"/\"version\": \"0.0.1\"/g)
Creating archive - Version 0.0.1...
Archive ver. 0.0.1 created!

Done.
```

Figure 18. Creation of Transportation BNA

2. Install the BNA in participating peer nodes (Figure 19):

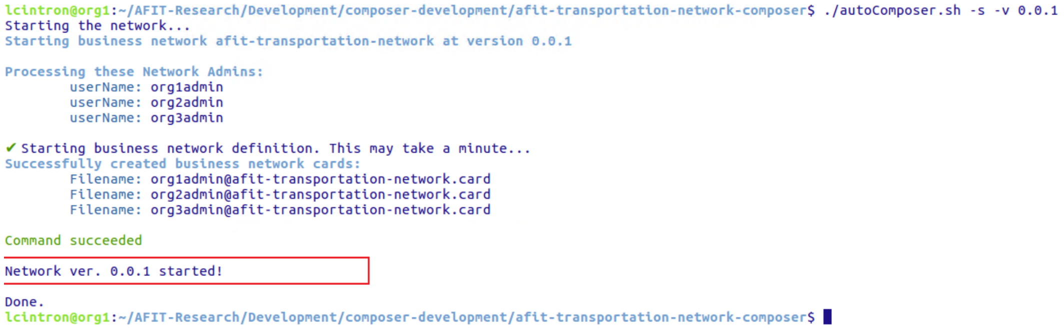


```
lcintron@org1:~/AFIT-Research/Development/composer-development/afit-transportation-network-composer$ ./autoComposer.sh -x -v 0.0.1
✔ Installing business network. This may take a minute...
Network ver. 0.0.1 installed!

Done.
```

Figure 19. Installation of Transportation BNA

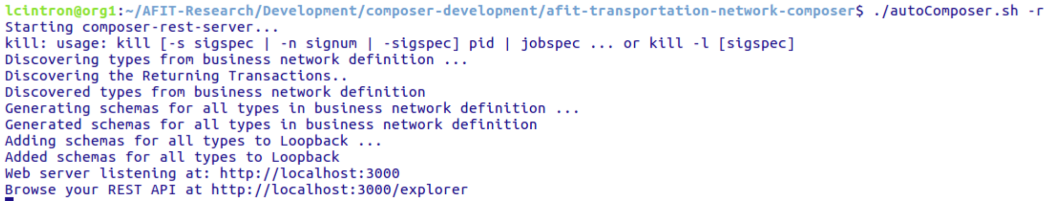
3. Start the Composer network from one of the peers with the BNA installed (Figure 20):

4. Start Composer-REST Server in all participating peers (Figure 21):



```
lcintron@org1:~/AFIT-Research/Development/composer-development/afit-transportation-network-composer$ ./autoComposer.sh -s -v 0.0.1
Starting the network...
Starting business network afit-transportation-network at version 0.0.1

Processing these Network Admins:
        userName: org1admin
        userName: org2admin
        userName: org3admin

✔ Starting business network definition. This may take a minute...
Successfully created business network cards:
        Filename: org1admin@afit-transportation-network.card
        Filename: org2admin@afit-transportation-network.card
        Filename: org3admin@afit-transportation-network.card

Command succeeded

Network ver. 0.0.1 started!

Done.
lcintron@org1:~/AFIT-Research/Development/composer-development/afit-transportation-network-composer$
```

Figure 20. Deployment of Transportation BNA



```
lcintron@org1:~/AFIT-Research/Development/composer-development/afit-transportation-network-composer$ ./autoComposer.sh -r
Starting composer-rest-server...
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
Discovering types from business network definition ...
Discovering the Returning Transactions..
Discovered types from business network definition
Generating schemas for all types in business network definition ...
Generated schemas for all types in business network definition
Adding schemas for all types to Loopback ...
Added schemas for all types to Loopback
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
```

Figure 21. Deployment of the REST Server using Hyperledger Composer Tools

3.9 Summary

This chapter described the proposed consortium for transportation services and the network architecture using a DLN it can rely on. It presented an analysis of strong DLTs alternatives and the framework selection criteria followed for the implementation of an experimental network. The frameworks and tools utilized to create, configure, and deploy the network are also discussed. LTMT, a tool for simulating a transportation environment with vehicles and RSUs is presented and described. Finally, the design of the HLF and HLC networks is introduced and the steps to deploy and initiate them are listed.

IV. Experimental Scenarios

4.1 Objective

The goal of this experiment is to model ITS infrastructure and applications using a DLN. Two specific applications are modeled in the HLF network. The first is a simple vehicle odometer mileage reporting and tampering identification function. The second is an accident data collection system. Results will be utilized to assess the effectiveness of the proposed network for collaborative distributed transportation services. Collected metrics will be analyzed to assess the overall performance of the network implementation under different loads within the same hardware platform. Modifications in network and code design parameters found to improve responsiveness and throughput are also discussed.

4.2 Assumptions

The assumptions made in this experiment are related to the simulation environment, HLF constraints and known factors, and experiment design decisions.

- Transaction Latency: Transaction times are measured from the time the client submits a transaction to the network for execution, to the time it is committed in the ledger and responds back to the client.

- Signal Loss: Wireless or wired signal losses are not modeled in the simulation environments.

- Vehicles and Sensors Authentication: RSUs and vehicles sending transactions are assumed to have the appropriate access rights to the services.

- Traffic Laws: Vehicles in the simulation do not perform stops at intersections or adjust their speed based on a road's speed limit. There's no concept of lanes in the simulation, just routes where vehicles move from their predefined source address to their destination address.

- Ambient Environment: There are no obstructions visual, or otherwise present in the simulation environment (i.e., the environment is clear with high visibility).

- Vehicles Only: No obstacles, except other vehicles in the same path, are present in the simulation environment (i.e., no pedestrians or wild animals).

- Event Data Recorder: Vehicles are equipped with data recording capabilities to store time, location, detected-vehicle ID, speed, heading, and misbehavior type, if any, of received BSMs.

- Vehicle Communications: Vehicles are equipped with means to establish a zero-latency data-link with other vehicles and roadside units, and can communicate with the HLF DLN services over the Internet. The inner workings of VANET communications as specified by the IEEE 1609 Family of Standards are not modeled for this experiment.

- Vehicle Misbehavior: Misbehavior is defined as a vehicle underspeeding (10 MPH below speed limit), overspeeding (10 MPH above speed limit), or stopped. The default speed limit in the simulation is 20 MPH.

- Vehicular Accidents: Vehicle collisions generated in LTMT are triggered when vehicles are detected within 5 meters of each other, at least one of the vehicles is misbehaving, and a pseudo-random accident event generator triggers an accident event. All other vehicles not involved in the accident continue moving toward their destination. In the modeled scenarios, it is assumed that one of the two vehicles involved in the accident reports it.

4.3 Control Variables

The control variables are those held constant over the execution of the various scenarios and experiments of this research, both at the simulation and DLN layers. These are listed in Table 6.

Table 6. Control Variables

Variable	Value	Description
Participating Organizations	3	Number of organizations in the network
Ordering Nodes	3	Number of HLF ordering service nodes in the network. One per Org.
Peer Nodes	3	Number of HLF peer nodes in the network. One per Org.
Channels	1	Number of blockchains in the DLN.
Zookeeper-Kafka Cluster	3 Zookeeper 4 Kafka Nodes	Zookeeper and Kafka nodes used for consensus.
World-State DB	CouchDB	Type of database software for maintaining the channel's world state. One per Organization.
TLS Communications	Disabled	TLS is disabled to ease up configuration and deployment of the network for multiple scenarios.
Number of Vehicles	3000	Number of vehicle records initialized in the DLN.
Number of RSUs	600	Number of RSU records initialized in the DLN.
Accident Probability	20%	Probability of an accident being triggered in LTMT when two vehicles have colliding locations and at least one misbehaving.
Speed Violation Threshold	10 MPH	Threshold of the current speed limit (+/-) for identifying speeding/underspeeding violations.
Transaction Arrival Rate	10-100 TX/s	Number of transactions sent to the within a 1 sec time period.

4.4 Independent Variables

Independent variables are specific to the scenarios aimed at evaluating and optimizing the performance of the network. These are listed and described in Table 7.

Table 7. Independent Variables

Variable	Units	Description
Transaction Arrival Rate	TPS	Transactions submitted per second
Block Size (*BatchSize*)	TPB	Maximum number of transactions per block
Block Timeout (*BatchTimeout*)	seconds	Maximum time to wait for a block to be filled
Endorsement Policy	-	Type of endorsement required to commit a transaction. Values: All-Orgs, 2-Of

4.5 Response Variables

The response variables for this research work are listed in Table 8. These are correlated to the performance of the network given a specific scenario. These values are measured at the application layer or gathered from the HLF network through the use of Hyperledger Explorer's API. Thorough analysis of these values will aid in determining the optimal configuration for the network given the hardware platform, deployed chaincode, and data models.

4.6 Application Scenarios

The next two subsections describe the use of the DLT network and services for two specific applications. The chosen applications can leverage the properties of blockchain features provided by HLF and HLC and take advantage of the collaborative, distributed, and immutable environment provided to identify possible fraudulent vehicle odometer rollbacks and invalid accident liability claims.

Table 8. Response Variables

Variable	Units	Description
Mean Transaction Success Rate	%	Percentage of successful transactions out of total submitted
Mean Transaction Throughput	TPS	Mean of transactions processed by the network, measured at the application layer
Peak Transaction Throughput	TPS	Peak number of transactions processed and committed, measured at the application layer
Peak Execution Rate	TPS	Peak number of transactions executed at and measured at the HLF layer
Peak Block Allocation	TPB	Peak number of transactions per block generated, measured at the HLF layer
Mean HLF Errors Detected	# Errors	Mean of HLF errors received at the application layer
Mean Connection Errors Detected	# Errors	Mean of other network errors received at the application layer
Average Response Time	seconds	Average of time to process a bulk of transactions, measured at the application layer

4.6.1 Vehicle Odometer Reading Reporting

Odometer fraud is the alteration of a vehicle's odometer with the goal of keeping it from displaying or recording the vehicle's actual total distance traveled. NHTSA estimates that odometer fraud due to tampering costs American car buyers over $1 billion annually due to over 450,000 vehicles sold each year with false odometer readings [44]. In the proposed consortium, participating vehicle services providers and automotive dealers can update vehicle records whenever these undergo maintenance or are checked at their facilities for routine inspections (e.g., smog tests in some states).

This simple scenario relies on the capability to submit vehicle maintenance reports which include a current odometer reading. By using a DLN chaincode operation or TPF, this value can be checked against the previously known odometer reading of the car and report back whether it has been tampered with if the value is less than previously reported. In addition, participating organizations can subscribe to events triggered by invalid odometer reports, to analyze or investigate the matter. Individuals in the process of purchasing a vehicle can verify that the odometer of the vehicle in question has not been tampered with by accessing services provided by their local Department of Motor Vehicles (DMV) or other businesses with access to the network. Given that DMV offices hold information about the owner of vehicles, any tampering detection information can be relayed to law enforcement or similar authorities (e.g., Office of Odometer Fraud Investigations at the USDOT) to initiate an investigation that could result in fraud charges.

This application can be easily employed in the deployed experimental network by modeling the following scenario. A vehicle service provider performs maintenance on a car and, upon completion, reports the maintenance performed along with the vehicle's odometer reading at that point in time to the consortium network, an action many shops and automotive dealerships already do through various private services (e.g., CARFAX). This odometer reading report can be validated against the last reported vehicle odometer reading. If the reading is valid, it is updated on the vehicle record. If it is invalid, the transaction is still committed to the blockchain, showing execution of the chaincode function and its result. However, the vehicle record is not updated and a notification is sent to channel subscribers. The network also responds back to the service provider with a valid or invalid response to its initial transaction. Organizations subscribed to the channel can automatically initiate an investigation about the matter or query the network for more evidence. The workflow of events for the scenario described is shown in Figure 22.

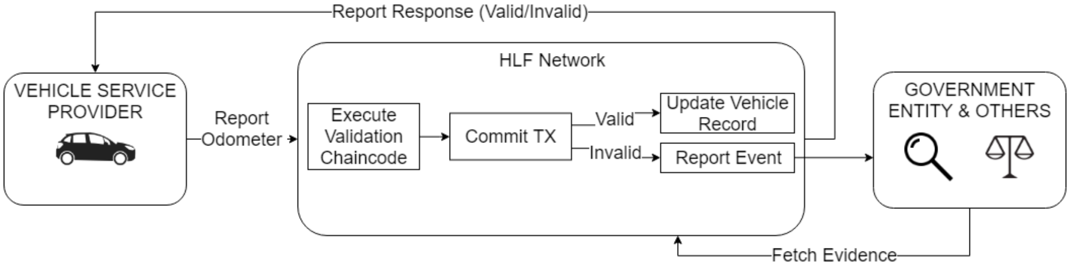


Figure 22. Workflow of Events for Odometer Tampering Detection

The TPF pseudo-code modeled for this scenario is shown in Algorithm 1. It does not perform any validation or corroboration with other data from the network. Actual TPF logic is included in Appendix F.

Algorithm 1 Vehicle Odometer Update Transaction

```
//$tx \leftarrow$ {odometer, vehicle}
if $tx.odometer > tx.vehicle.odometer$ then
    $tx.vehicle.odometer \leftarrow tx.odometer$
    $assetRegistry \leftarrow GetVehicleRegistry()$
    $assetRegistry.update(tx.vehicle)$
    $tx.valid \leftarrow true$
else
    $tx.valid \leftarrow false$
    emit(InvalidOdometerReadingEvent)
end if
return $tx$
```

4.6.2 Accident Data Collection

Accident Fraud is another common problem vehicle owners and transportation entities face every day. A 2012 study shows that common fraudulent activities which include false accident claims, staged accidents, and false liability disputes, have cost insurance companies between $5.6 to $7.7 billion in excess payments [45]. Another study from 2016 showed that nearly 75% of insurance companies have opted to introduce automated systems to detect false claims and reduce their losses in excess payments [46]. The proposed DLN can provide false claim detection capabilities and accident liability evidence for purposes of attribution through the corroboration of accident data recorded and submitted by vehicles involved and witnessing the event of an accident.

This scenario focuses on the storage of accident event reports and witnessed data recorded by vehicles and road sensors to create a snapshot of events within a window of time before an accident and serve as evidence in the identification of liable parties, an extension of the similar approach in [29]. Each of the vehicles in the simulation share information with other vehicles and infrastructure through V2V/V2I communication channels, and with the transportation DLN (Figure 14) via a wireless network Internet access card as depicted in Figure 23.

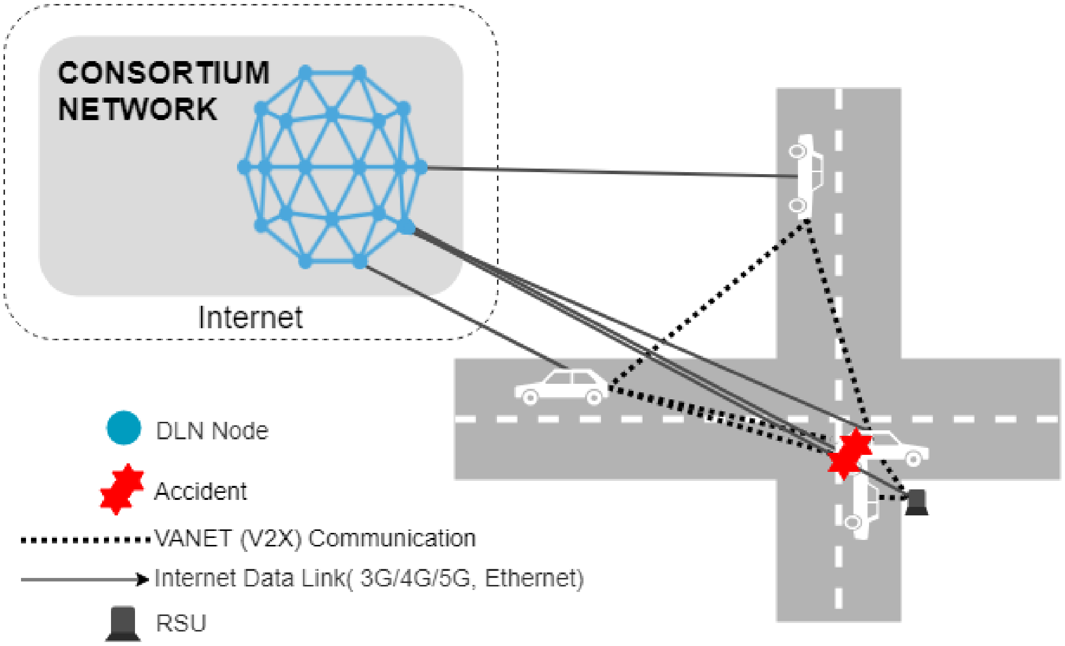

CONSORTIUM NETWORK
Internet
DLN Node
Accident
VANET (V2X) Communication
Internet Data Link(3G/4G/5G, Ethernet)
RSU

Figure 23. Operational View of ITS Infrastructure with HLF

Simulation scenarios consist of a predefined number of vehicles and RSUs in a specified area. Vehicles are assigned a point of origin and a destination and move until they have arrived at their destination. For this study, RSUs only collect data on misbehaving vehicles. Vehicles broadcast BSMs as they move. When the simulation triggers an accident between two vehicles within collision distance, one of the involved vehicles reports the event to the DLN, along with its observed and logged OBU data as well as IDs of other vehicles involved. The DLN then validates the source and other involved vehicles, creates an Accident Event report, and notifies the application that the report has been submitted. At this point the application notifies all vehicles in the area and requests that they report all witnessed data within the location of the accident during the time frame of the accident or any witnessed misbehavior data for any of the vehicles involved. The workflow of the steps described for this scenario are depicted by Figure 24.

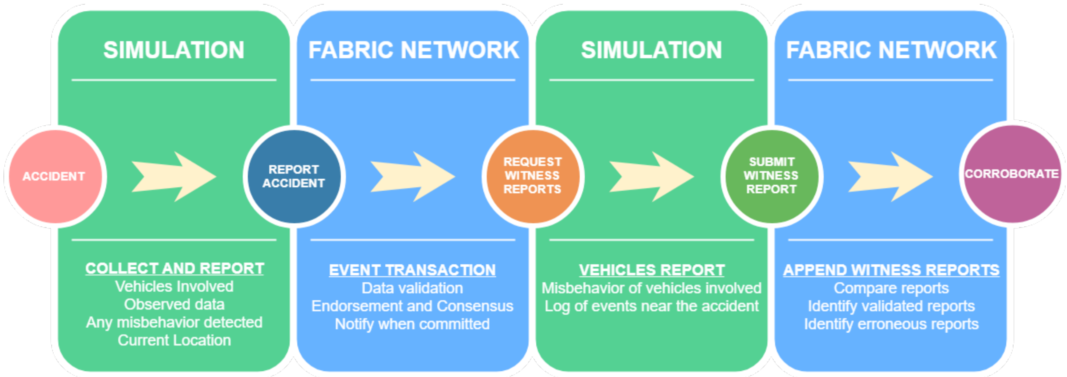


Figure 24. Workflow of Accident Event Scenario

The pseudo-code in Algorithm 2 describes the approach for creating a road event report in the network. Triggered by either of the two vehicles involved in the accident, the submitted transaction contains the IDs of the vehicles involved, the time of the accident, location, and a list of witnessed events from the perspective of the reporting vehicle (e.g., misbehavior, detected speeds, changes in location). The transaction is then executed to verify the existence of the sensors and validity of the inputs. If the parameters of the transaction are valid, a road event entry containing the initial information reported by the originating vehicle is created and a reference to it is added to the vehicles involved in the accident. A valid accident event entry is created and emitted. Invalid accident report transactions are recorded in the blockchain, however, no assets are updated with a reference to this invalid transaction. However, records of invalid events like this could be later used to identify network spoofers.

Algorithm 2 Road Event Transaction

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

In the simulation, once confirmation of the accident report creation has been received, all vehicles and RSUs are notified about the accident event, and these report back to the network all witnessed misbehavior detected from BSMs of the vehicles involved in the accident. Vehicles that had been within 100 meters of the accident also report back misbehavior and a log of all BSMs received 10 seconds prior to the accident occurring. The pseudo-code shown in Algorithm 3 shows a specified road event is updated with witnessed data. The TPF executing this transaction ensures that reporting vehicles are valid and have been in the vicinity of the accident.

Algorithm 3 Road Event Witnessed Data Transaction

```
//tx $\leftarrow$ {sourceId, eventId, WitnessedData}
if tx.WitnessedData.length$>$ 0 & SensorExists(tx.sourceId) & IsValid(tx) then
    $wd \leftarrow WitnessedData(tx)$
    $WitnessedDataRegistry.add(wd)$
    emit(WitnessedDataSubmitted)
else
    emit(InvalidWitnessedDataTx)
end if
```

As previously mentioned, the aggregation of collected accident data can help to identify liable parties in the event of an accident. Furthermore, this data can be corroborated in the distributed network to identify any potentially false or misleading reports submitted by dishonest sensors. The pseudo code for a TPF that can accomplish this task is shown in Algorithm 4. It describes how the network can validate an event's witnessed data to ensure consensus on whether or not evidence exists showing the described events did indeed happen (i.e., same observed behavior by multiple unrelated parties) and perhaps identify potential misbehavior. This TPF can be triggered by a consortium entity looking into the event (e.g., insurance company, law enforcement) or automatically after a specified period of time.

Further analysis and experimentation for the corroboration of accident data and its use in accident forensics is outside of the scope of this research due to time limitations.

4.7 Performance Evaluation

Characterizing the performance of the experimental network is important even in platforms with constrained resources like the one utilized for this experiment. Doing so provides a clear picture of the type of application scenarios that can be simulated in similar environments. Additionally, it helps with finding the optimal network configuration parameters to get the best performance out of the network at hand. This evaluation aims to characterize the experimental network as means

Algorithm 4 Corroborate Witnessed Data for a Specific Road Event

```
re ← {RoadEvent}
for all WitnessReport $wr$ ∈ RoadEvent $re$ do
    $possibleValidators \leftarrow getWitnessReports(\Delta T, wr,$ WitnessReports $\in re)$
    for all WitnessReport $p \in possibleValidators$ do
        if $p$ validates $observed$ then
            $observed.validatedBy \leftarrow observed.validatedBy \cup p.id$
        end if
        if $p.seenInRange(observed)$ then
            $observed.seenBy \leftarrow observed.seenBy \cup p.id$
        end if
    end for
end for
$validated \leftarrow getValidatedReports()$
$seen \leftarrow getSeenVehicleReporters()$
$possibleSpoofers \leftarrow re.WitnessReports - (validated \cap seen)$
```

to identify the constraints for other simulations that could be performed. Results of this evaluation are not a measure of HLC/HLC peak performance but the overall performance of the network given the selected HLF network configuration, HLC TPF, and the API layer provided by HLC. Works such as [37] have shown HLF's ability to reach a transaction throughput of ∼2.5K TPS, however, no documented research showing the performance of both HLC and HLF technologies integrated has been found.

The performance of the network is evaluated by submitting various bulk transactions of the same type and analyzing the response variables resulting from those inputs, a similar approach to that performed in [37]. The workflow for the experiment is shown in Figure 25 and consists of stimulating the network with transactions, fetching committed transactions and block information such as the creation timestamp and transaction count, and analyzing the data by calculating the response variables listed in Section 4.5. The script utilized for the execution of this workflow is found in Appendix J. Three HLF network configurations are tested, each with an All-Orgs and the 2-Of endorsement policy previously shown in Figure 16; these are detailed in Table 9. All transactions are submitted to peer nodes in a round-robin fashion via HTTP POST request to their REST API endpoints to create a new *RoadEvent* record. The HTTP POST requests consist of a payload of 309 bytes. The vehicle IDs used for the transactions are incremented to avoid MVCC errors due to multiple accident events within the same time frame involving the same vehicles. The network was re-initialized (all data discarded) for each change in configuration, requiring re-installation of the HLC BNA and re-initialization of the HLC REST and Hyperledger Explorer servers.

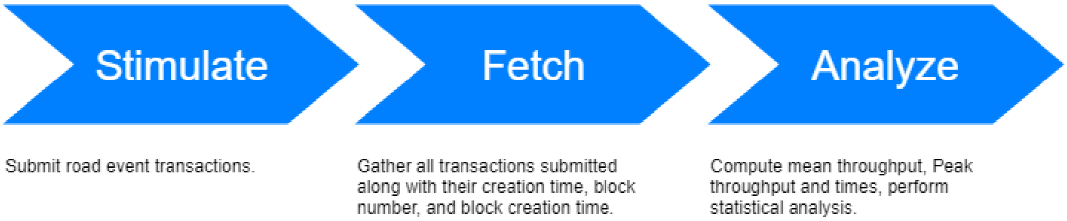

Stimulate
Fetch
Analyze
Submit road event transactions.
Gather all transactions submitted along with their creation time, block number, and block creation time.
Compute mean throughput, Peak throughput and times, perform statistical analysis.

Figure 25. Workflow of Performance Analysis Experiment

Network Configuration 1 is the default block/timeout/endorsement configuration that ships with HLF and HLC. The default endorsement policy requires a member of each participating organization to execute a given transaction and achieve the same result as the originating node. Another endorsement policy is called type 2-Of. The 2-Of endorsement policy for testing is the only other possible policy in this setup without completely disabling endorsement (a security feature in HLF). Based on observations discussed in [37] the block timeout for all other configurations tested is reduced to 1 second. A 1 second block timeout allows for transactions submitted during times where the arrival rate is less than the block size to be allocated into a block without having to wait too long before being sent for validation and committal. This value does not include the time required for broadcast, validation, and committal.

Table 9. Fabric Network Configurations for Performance Analysis.

Configuration	Block Size	Block Timeout	Endorsement Policy
1	10 TPB	2 s	All Orgs
2	10 TPB	1 s	2-Of(Org1, Org2, Org3)
3	50 TPB	1 s	All Orgs
4	50 TPB	1 s	2-Of(Org1, Org2, Org3)
5	100 TPB	1 s	All Orgs
6	100 TPB	1 s	2-Of(Org1, Org2, Org3)

The workflow described in Figure 25 is enabled through a series of scripts that automate all three steps of the process for each block/endorsement configuration. No other applications or tools are open on the host computer while the workflow scripts are running to minimize performance degradation of the VMs. The final output of the scripted workflow is analyzed across the different configurations.

4.7.1 Test Matrix

Table 10 is an excerpt from the test matrix in Appendix I for this experiment. For each configuration shown in Table 9, 10 different arrival rates batches are submitted three times. As a result, the experiment takes 180 test runs (treatments).

4.7.2 Data Collection

Performance and network stimulation data is recorded with scripts that generate the transaction and aggregated to the data queried from the blockchain network using Hyperledger Explorer's API. In other words, metrics to assess the performance of the network are measured at both the application (stimulation) and DLN(result) layers. Collected data is then analyzed and exported into a Comma-Separated Values (CSV) file that can be opened with Microsoft Excel for inspection and further analysis. A sample output file from the 3-step process aforementioned is included in Appendix K.

Table 10. Excerpt from Test Matrix

Arrival Rate (TX)	Endorsement Policy	Block Configuration
10	All Orgs	10 TPB - 2 s
20	All Orgs	10 TPB - 2 s
30	All Orgs	10 TPB - 2 s
40	All Orgs	10 TPB - 2 s
50	All Orgs	10 TPB - 2 s
60	All Orgs	10 TPB - 2 s
70	All Orgs	10 TPB - 2 s
80	All Orgs	10 TPB - 2 s
90	All Orgs	10 TPB - 2 s
100	All Orgs	10 TPB - 2 s
10	2-Of	10 TPB - 1 s
20	2-Of	10 TPB - 1 s
30	2-Of	10 TPB - 1 s
40	2-Of	10 TPB - 1 s
50	2-Of	10 TPB - 1 s
60	2-Of	10 TPB - 1 s
70	2-Of	10 TPB - 1 s
80	2-Of	10 TPB - 1 s
90	2-Of	10 TPB - 1 s
100	2-Of	10 TPB - 1 s

4.7.3 Analysis

An instance of the Julia programming environment is used to perform calculations from the data obtained from Hyperledger Explorer. These include mean transactions per block (block allocation), total blocks created, and peak block throughput (see Appendix L). The final output of all scenarios is analyzed to determine whether the differences in network parameters cause any statistically significant improvement.

A Mann-Whitney U Test is a non-parametric test performed to determine the statistical significance of the difference in mean throughput between the baseline configuration (Configuration 1) and the rest of the configurations with a 95% significance level. The use of a Mann-Whitney U Test is adequate for this analysis since it tests for the difference between two groups on a single variable independent of the distribution, often used when the sample is small and the data is ordinal [47]. The resulting U-Value describes the degree of overlap between two groups. U-Values below the Critical U-Value for $p < 0.05$ mean that there exist significant differences between the two groups, hence rejecting the null hypothesis (i.e., there are differences between the means of the two groups). U-values greater than the Critical U-Value result in failing to reject the null hypothesis.

4.7.4 Tools

The tools in Table 11 were used to stimulate the network, query data from the blockchain, and analyze the performance metrics of the HLF/HLC network. The scripts for submitting the road event request as well as the request related response factors were written in JavaScript and executed using NodeJS.

Table 11. Data Gathering and Analysis Tools.

Name	Version	Description
NodeJS	v8.11.3	JavaScript run-time built on Chrome's V8 JavaScript engine.
Julia	v1.0.3	Dynamic programming language with numerical analysis and data visualization capabilities.
Genie.jl	v0.1	Library for web server development in Julia.

Once the network was stimulated, a separate script queried Hyperledger Explorer for all blocks and transactions created since the scenario started. The data obtained from Hyperledger Explorer is then analyzed using the Julia environment through the grouping and calculation of transaction/block related response variables (see Appendix J and Appendix L). The resulting file is then imported into an Excel spreadsheet for plot generation and further analysis. The Genie.jl module for Julia was utilized to avoid re-initialization of the Julia kernel for every command-line call to process the data file from the previous step.

4.8 Summary

This chapter described the experimental objects, assumptions, and variables. It presented the use of a DLN for two applications that help mitigate problems costing transportation-related organizations and vehicle owners alike millions of dollars every year: odometer fraud protection and accident data collection for vehicular forensics. It then described the approach for evaluating the performance of the experimental network implementation which entails the stimulation of the network using *RoadEvent* transactions and accessing the blockchain to gather data that will be used to measure its performance. The workflow and tools used for the performance analysis are also discussed.

V. Observations and Analysis

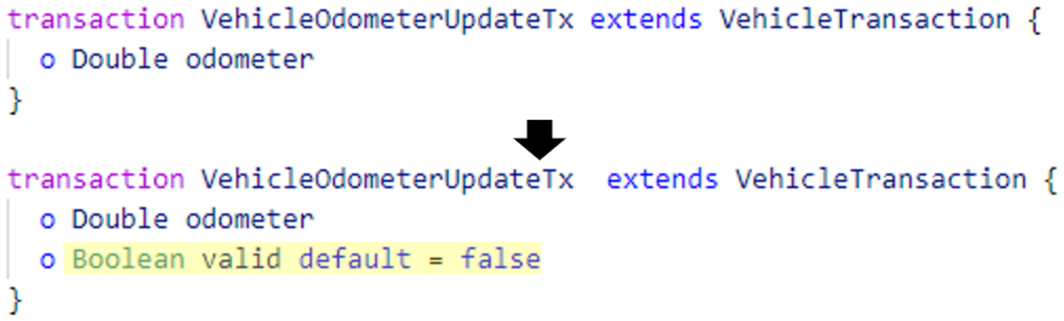
5.1 Overview

This chapter presents the observations, results, and analysis from the experimental activities described in Chapters III and IV. Issues found during the execution of distributed applications are discussed and respective solutions are described. Findings and performance metrics obtained through load testing the network under various configurations are presented and compared among each other to show the statistical significance of the differences in terms of mean transaction throughput. Possible sources of error for the findings are discussed. Finally, it discusses benefits, drawbacks and challenges, and security and privacy concerns with the experimental network implementation.

5.2 Application Scenarios - Results and Model Enhancements

5.2.1 Odometer Reading Report Scenario

The odometer-reading records-keeping scenario was executed successfully. As expected, attempts to update a vehicle's odometer record with a value less than the latest record in the blockchain did not result in a change to the vehicle's state and all other organizations received an event notification of the attempt to make the invalid change. The definition for the odometer update transaction had to be modified to include a field denoting whether it is a valid update or not (noted in Algorithm 1 and shown in Figure 26). Since TPFs return the transaction object submitted for processing along with any updates made to it (e.g., transaction ID, timestamp), this change allows the entity submitting the request to know whether the value was updated successfully or not without having to listen for the corresponding channel event.



```
transaction VehicleOdometerUpdateTx extends VehicleTransaction {
  o Double odometer
}

transaction VehicleOdometerUpdateTx  extends VehicleTransaction {
  o Double odometer
  o Boolean valid default = false
}
```

Figure 26. ***VehicleOdometerUpdate*** **Transaction Definition Change**

An operational implementation for the same activity requires a way to roll-back odometer records for vehicles that may have had the engine swapped or their odometer reading records has been incorrectly updated. A verification workflow for this scenario can be implemented in the HLF DLN requiring multiple administrators or certified validators (e.g., government agencies) to sign off on the roll-back transaction. In the event of fraud detection, the identity of the individual responsible can be obtained and he or she could be held accountable.

5.2.2 Accident Reports Scenario

Initial simulations of the accident data collection scenario using the TPF described by Algorithm 2 resulted in successful generation of road events. Conversely, many accident witness data transactions for the road events created were reported as invalid by the DLN. The results for these simulations are shown in Table 12. The calculated success rates for the witness report transactions are shown in Figure 27, showing an average success rate of 29.2%. The difference in success rates among the five scenarios is due to the varying amount of witness reports per accident generated in the simulation.

The unsuccessful witness data report transactions returned the following error:

Table 12. Accident Event Witness Report Results (Initial Data Model Definition)

Run	Accident Events	Witness Reports	Total Successful TX	Total MVCC Errors
1	4	317	100	221
2	2	121	32	91
3	2	185	50	137
4	7	651	217	441
5	4	441	122	323

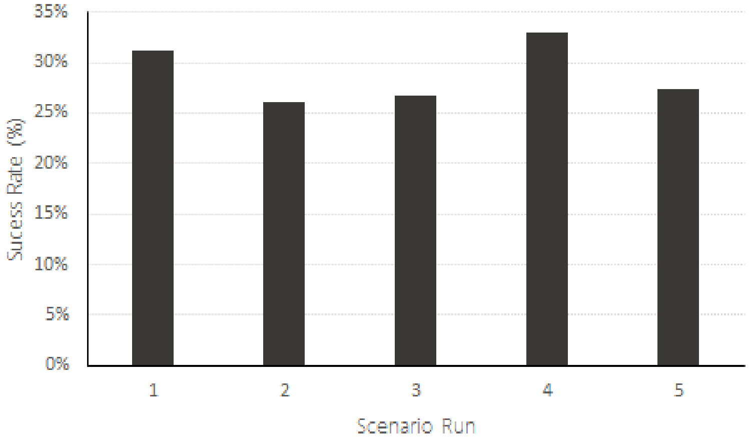
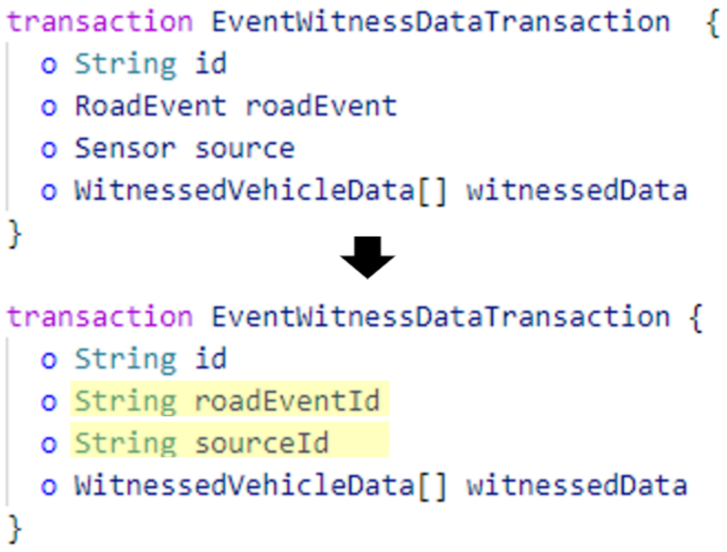


Figure 27. WitnessData Report TX Success Rate with Initial Data Model Definitions

Error trying invoke chaincode. Error: Peer has rejected transaction with code MVCC_READ_CONFLICT. This is due to HLF's employment of MVCC. MVCC validation takes place at the peers during the commit phase of a transaction. It ensures that the state of objects to be read or written during commit are at the same state they were when the transaction was endorsed during the execution phase [37, 48, 39]. Transactions that trigger an MVCC error are still recorded in the ledger but their result does not affect global state, in other words, it becomes a void transaction.

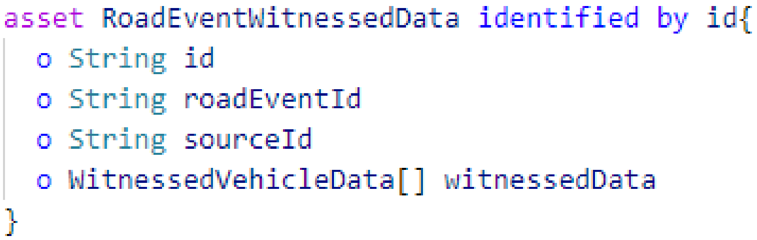
Since the fast rate of updates to a specific road event asset with witness data from all vehicles with observed BSM parameters/road-behavior causes the error, 100% of the errors were eliminated by creating a new object for each witness report instead of updating the same *RoadEvent* object's witness reports attribute. The witness report *transaction* and *asset* model definitions contain the ID of the road event object they provide evidence for and the ID of the source sensor as a *String* instead of a reference to the object that gets front loaded during the execution phase of the transaction. The transaction definition change described is shown in Figure 28 and executing the *EventWitnessedDataReport* TPF with such a transaction results in the creation of a new entry of type *RoadEventWitnessedData* with the definition shown in Figure 29. The *WitnessedVehicleData* array was removed from the *RoadEvent* asset definition since it is no longer needed. This solution is analogous to creating a new table for witness reports in a relational database system with a foreign key relationship (one-to-many) mapping to the road event entry at a different table. When witness reports for a specific road event need to be analyzed or used for further processing (e.g., corroboration), these can be fetched using HLC queries which follow a similar format to basic T-SQL queries.



```
transaction EventWitnessDataTransaction  {
  o String id
  o RoadEvent roadEvent
  o Sensor source
  o WitnessedVehicleData[] witnessedData
}
```

```
transaction EventWitnessDataTransaction {
  o String id
  o String roadEventId
  o String sourceId
  o WitnessedVehicleData[] witnessedData
}
```

Figure 28. ***EventWitnessData*** **Transaction Definition Change**



```
asset RoadEventWitnessedData identified by id{
  o String id
  o String roadEventId
  o String sourceId
  o WitnessedVehicleData[] witnessedData
}
```

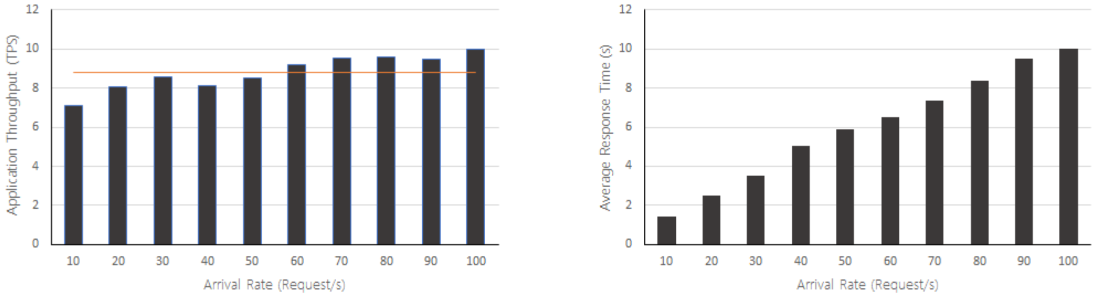
Figure 29. ***RoadEventWitnessedData*** **Asset Definition**

A TPF implementation of the pseudo-code described by Algorithm 4 executed against road events with witness identified valid reports and vehicles observed by other vehicles involved or reporting accident information. However, it is not a comprehensive implementation for data corroboration and was not validated as part of this research. Accurate corroboration of accident events using reported data collected by systems or applications like the one presented here is outside the scope of this research and only identified to illustrate future capabilities that could be implemented.

5.3 Performance Experiment Results

Batches of 10 to 100 road event transactions in increments of 10 were submitted within a 1 second time period. As shown in Figure 30(a), load testing the network with the default configuration (10 TPB, 2 second block timeout, endorsed by all organizations), Configuration 1, an average transaction throughput of 8.82 TPS and an average response time of 6.01 seconds were measured at the application layer with a transaction arrival rates ranging from 10 to 100 TPS. Figure 30(b) shows that the average response time increased linearly with the increase in the arrival rate for the scenarios.

Figure 31 shows the average network throughput of each configuration given increasing transaction arrival rates. At first glance, it can be observed that the con-



(a) Average Transaction Throughput

(b) Average Response Time

Figure 30. Network Performance of Default Configuration

figurations with a 2-Of type endorsement resulted in higher throughput values than the configurations requiring endorsement by all organizations in the network. This is expected because nodes need not wait for the two other peer nodes to have the transaction ready for blocking. Instead, they only require endorsement by one other peer node reducing the transaction execution latency. Moreover, transactions with fewer endorsement signatures require less time at the validation phase. Another observation is that the average throughput of configurations with the default endorsement policy increased almost linearly with increasing arrival rates. Conversely, configurations with 2-Of endorsement show a faster throughput growth with arrival rates between 10 and 60 TPS and a slim growth after 60 TPS.

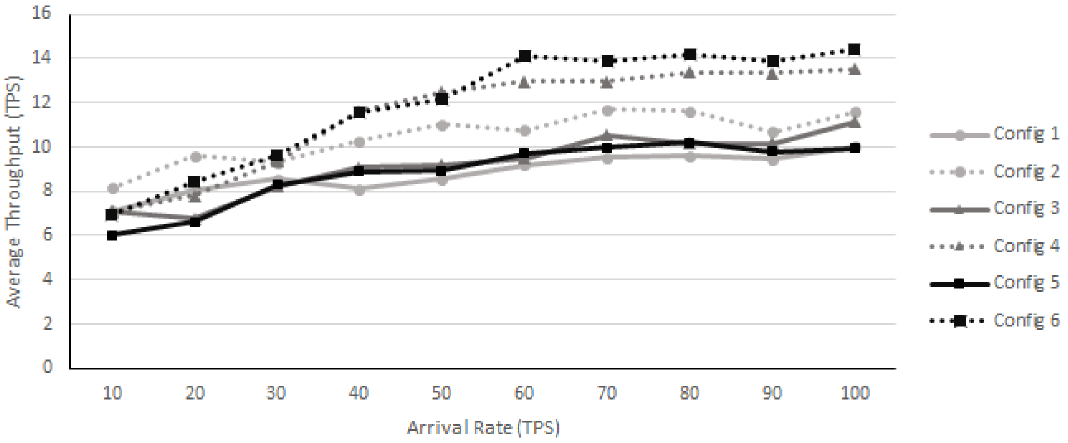


Figure 31. Average Throughput of Configurations

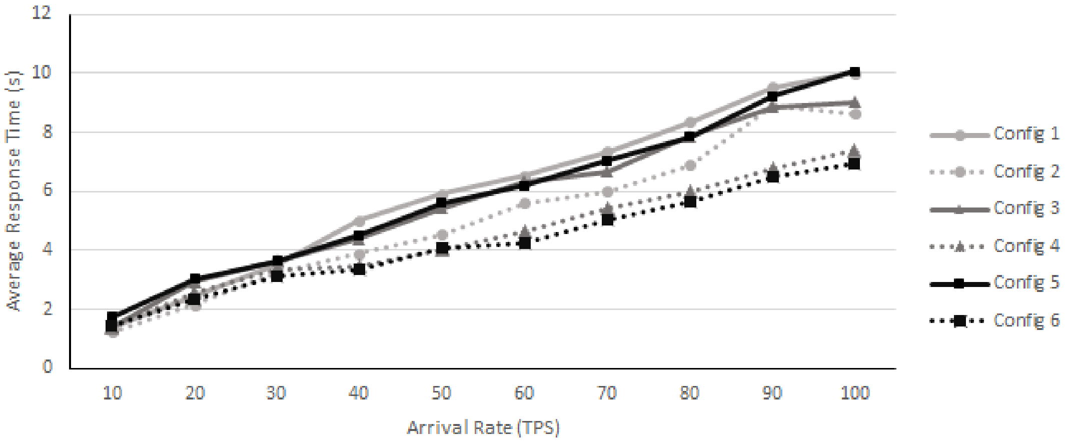


Figure 32. Average Response Time of Configurations

Another observation based on the data shown in Figure 31 is that most configurations have their best performance when the arrival rates are near their max allowable transaction allocation per block. At an arrival rate of 10 TPS, Configurations 1 and 2 performed better than all other configurations with the same endorsement, averaging 7.12 TPS and 8.14 TPS respectively. Similarly, at an arrival rate of 50 TPS, Configurations 3 and 4 outperformed all other configurations of the same endorsement with throughputs averaging 9.21 TPS and 12.49 TPS respectively. Conversely, at an arrival rate of 100 TPS, only Configuration 6 outperformed all other configurations with the same endorsement policy. Configuration 5, however, did not outperform all other configurations at an arrival rate of 100 TPS. This is due to all blocks generated with the 100 TPB max block allocation configuration being triggered after a timeout and not by filling the blocks since the peak block allocation for all configurations is 58 TPB, shown in Figure 33. This resulted in Configuration 3 having better performance than Configuration 5 (All-Orgs Endorsement) at arrival rates greater than 50 TPS. Another important observation from Figure 34, is that the peak execution rate of all configurations increased almost linearly at a slightly lower rate than the arrival rate, maxing out at 80 TPS. Based on these results, it can be inferred that the latency due to transaction endorsement request/execution in addition to the latency for endorsement signature validation far surpasses the latency introduced by block timeouts.

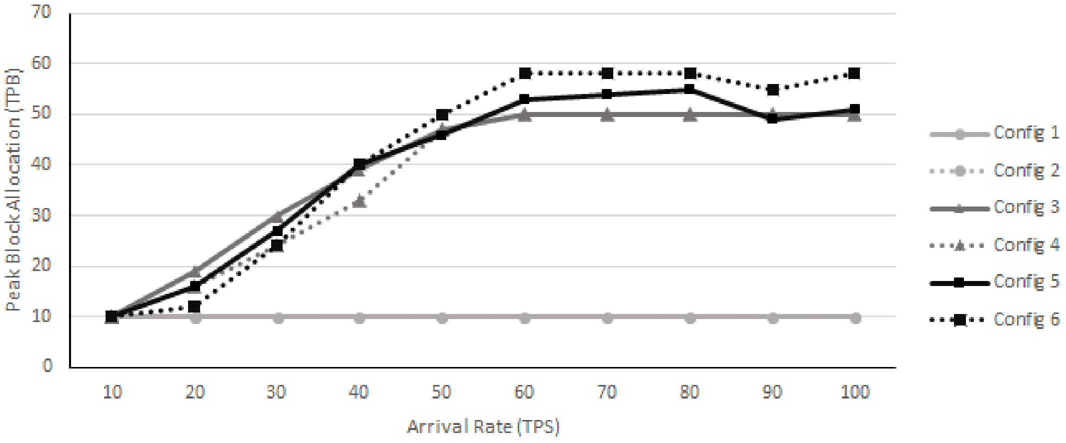


Figure 33. Peak Block Allocations of Configurations

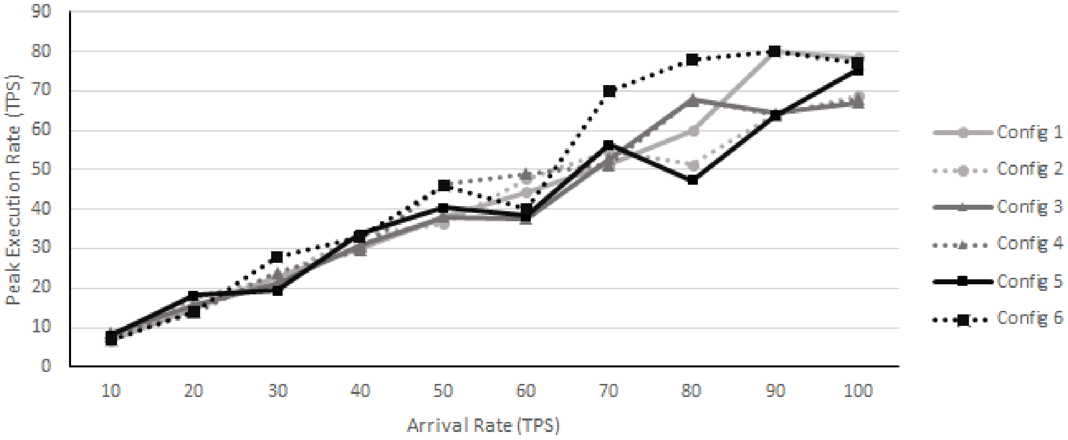


Figure 34. Peak Execution Rates of Configurations

The results of the Mann-Whitney tests conducted to identify the significance of the differences in mean between the baseline configuration and all other ones are shown in Table 13. The only significant differences in mean transaction throughput arise from the 2-Of endorsement policy.

Based on previous analysis, Configuration 6 had the best overall performance

Table 13. Mann-Whitney Test Results

Configurations	U-Value	Critical Value at $p < 0.05$	Significance at $p < 0.05$
1 vs 2	12	27	Significant
1 vs 3	39	27	Not Significant
1 vs 4	23	27	Significant
1 vs 5	41	27	Not Significant
1 vs 6	18	27	Significant

with the lowest mean response time and highest mean throughput with arrival rates ranging from 10 to 100 TPS. It yielded a peak throughput of 14.77 TPS with a 41.5% improvement over the default configuration (Configuration 1). Its improvement over configurations with the same endorsement policy, however, is not significant.

The overall throughput of the network can be improved by tailoring the configuration to the applications that utilize the specific channel. As the variety of applications employing the use of the transportation DLN increases, the transaction throughput of the network as a whole can be increased and the latency decreased by adding more channels to support those applications [37]. This experiment details considerations that must be accounted for when looking to improve an HLF DLN.

5.3.1 Possible Sources of Errors

There are many tools and layer implementations that were configured to enable the experimental network. Consequently, there are many possible factors that may have impacted the precision or accuracy of the response variables gathered for this experiment. The possible sources of error for the response variables are identified and described below:

- *Precision of block and transaction timestamps in Hyperledger Explorer.* Metrics gathered through Hyperledger Explorers API show blocks and transaction timestamps with up to 1 second precision. This prevents the calculation of precise Blocks Per Second (BPS) and TPS (measured at the DLN layer) factors. As a result, TPS metrics are measure at the application layer and only peak BPS values are reported

- *Virtualization of nodes and network interface cards.* The virtualized network is prone to processing and memory related interruptions due to the limited amount of processor threads and memory available to the VMs and which are also shared with the host operating system as scenarios are simulated.

- *Latencies introduced by middleware.* There is insufficient data to determine the root cause of the delays seen during load-testing. However, given the high number of transactions being received and executed by HLF (based on their timestamps), it is possible that these are introduced by the HLC services managing the REST API and the network TPFs. It is also known that using CouchDB as the state database for HLF results in slower transaction throughput [37], thus the implementation of this database in a hardware constrained environment may have magnified its effects on the overall performance.

5.4 Benefits

One of the main benefits of having a DLN is having access to a pseudo-immutable ledger that contains records of transactions and world-state changes in a cryptographically secured way. This network provides native auditing services that could prove crucial in ensuring the reliability and safety of modern transportation infrastructure. Other benefits of this implementation are listed below.

- *Event Attestation.* Authentication and corroboration mechanisms of the consortium model for ITSs enable attestation of the events stored in the ledger. Although not a requirement for all transactions, the implementation of the architecture can define which transactions need endorsement at the services layer and how to corroborate transportation event data stored in the ledger.

- *Auditability.* The DLN keeps an auditable history of valid and invalid transactions submitted by transportation systems, vehicles, RSUs, and sensors as well as individuals representing government organizations, and other participating stakeholders [49]. The trusted, comprehensive, and time-based log of events can be used not only to provide services for ongoing operations of the smart transportation ecosystem but to confirm (or deny) claims involving smart transportation systems and services (e.g., law enforcement).

- *Decentralized Execution.* Unlike centralized transactions systems, DLN peers can communicate without relying on a central information system or entity [25]. This reduces the dependency on third-party services (other than those used for platform hosting) for the DLN to operate. If at any point in time, a participating organization decides to leave the consortium, it can do so without affecting the availability of the network or the data stored in other peer ledgers. In addition, downtime or degraded performance of individual nodes does not result in the same effect on the network.

- *DDoS Attacks Resiliency.* Distributed Denial of Service Attacks (DDoS) that rely on spoofing or generation of bogus fake transactions can be mitigated by the membership and authentication features in the services layer and the network's decentralized architecture. When implemented at a large scale (e.g., across a large number of participating nodes), these attacks can be detrimental to the performance of any network. In a consortium DLN, all participating users/nodes must be authenticated thereby thwarting an attack by immediately revoking access to the specific user(s). Detection and access revocation is a feature that can be implemented and automated using a smart-contract. As a result, DDoS attacks on a subset of the network does not necessarily result in downtime for the whole network.

- *Sybil Attacks Resiliency.* A Sybil attack occurs when a single entity impersonates multiple other entities to create fake transactions or events [50]. Since users in the DLN are authenticated using PKI, an attacker would need the private keys of the members to impersonate. Similar to preventing a DDoS attack, the proposed platform can include contract configurations that allow the automated detection of impersonation of other users or entities via the corroboration and validation.

- *Zero Anonymity.* Since all users of the DLN are authenticated, all participants are known to the consortium entities (each entity controls its membership services). As a result, all transactions are authenticated and the identities of the transaction originators are known. Thus, members can be held accountable for their misbehavior.

- *Privacy.* Permissioned networks allow control over who gets access to the data in the ledger(s) maintained by the network. This means that users and entities grant read/write rights to users/devices on the network. For example, an organization may have access to traffic events on a specific region but no personally identifiable information.

HLF's modularity makes it an attractive framework for implementing DLNs that can interface with available infrastructure and services. By resolving transactions into a world-state database, stakeholders can execute rich queries to gather the data needed given they have the proper access rights to read it. Additionally, as a permissioned network framework, its implementations ensure zero anonymity of the transactions; all identities on the DLN are authenticated, and all participants are known to the consortium entities (each entity controls its membership services). As a result of transactions being signed by the originators and corroborators and later validated before being committed to the ledger, participants can be held accountable for their misbehavior. Issues concerning the protection of vehicular identities for preserving the privacy of vehicle owners are discussed in section 5.6.

5.5 Drawbacks & Challenges

5.5.1 Documentation

A key challenge in this research work has been finding detailed documentation on HLF and HLC configuration parameters and tools. These two projects host their documentation online, including tutorials and command/architecture references. Yet many concepts still lack details that would allow developers and network architects to both understand the capabilities provided and instantiate a network under optimal conditions.

5.5.2 BFT vs. CFT in Fabric

This implementation, although crash-fault tolerant through Apache Kafka, it is not BFT. Byzantine faults refer to faulty nodes that may appear fully functional but may produce inconsistent results either maliciously or unintended. For it to be resilient against Byzantine Faults, the ordering nodes must implement a different consensus algorithm. Sousa et al. in [51] developed a BFT consensus module for HLF called BFT-SMART with a tentative execution of requests approach similar to the PBFT approach depicted in [26]. This module is not included as part of HLF and its performance and reliability in production environments are still unknown.

5.5.3 High Storage Volume Requirements

Given the large amount of data transacted in transportation environments, storage may be an issue of concern. Once recorded in the blockchain ledger, transactions in these networks cannot be erased or tampered with, so this ledger proliferates quickly. As a result, peer nodes must be allocated large storage volumes to accommodate for this data, resulting in high hosting costs over time for all participants. Consequently, nodes who experience downtime will suffer long synchronization times which could extend endorsement downtime and lead to transaction execution failures. Lastly, framework components such as those provided by Hyperledger Composer are relatively immature, and therefore suffer from issues that often result in unresponsiveness during times with high transaction arrival rates, as experienced in this work.

5.6 Security & Privacy

PKI services and access control rules allow secure access to data by privileged users as defined by the consortium. Implementations can allow for users (e.g., vehicle owners) to control who is able to see data involving his or her vehicle. VPKI as defined in IEEE 1609.2 can be supported in a HLF DLN implementation by having the Vehicle Certificate Manager services integrated into the MSPs. Since VPKI relies on providing pseudonymity to vehicles, such certificates can be utilized to authenticate vehicles or sign transaction data thereby increasing the trust in the source of the data without revealing a source identity. However, authorities can identify the real identity of a vehicle pseudonym in case of accidents or legal investigations [52]. The chosen framework allows for revocation of certificates preventing participants from unauthorized access once their credentials have been revoked making it suitable for the integration of VPKI services.

5.6.1 Ordering Service Nodes - Potentially Vulnerable

HLF OSNs, although not involved in checking transactions, could be compromised and used to gain access to all transactions received, or distributed by the Kafka cluster. These nodes can be manipulated to intercept transactions sent and received by the ordering service. If information privacy is required (e.g., PII), HLF provides Private Data channels that create a separate private ledger among parties. Private channel transactions are not sent to the orderer, instead, a hash of the transaction and their timestamp are sent, thereby keeping an orderer from being able to observe a transaction's content. Enabling Transport Layer Security (TLS) communications can prevent man-in-the-middle attacks involving OSNs.

5.6.2 Issues with Zookeeper-Kafka Cluster

One downside of HLF's Zookeeper-Kafka consensus is the fact that it adds a form of centralization to the network. In truly decentralized networks, nodes are not dependent on a service to continue operation, however, in HLF networks orderers depend on the Zookeeper-Kafka cluster's availability to operate. This means that in a consortium environment, it is in the best interest of the participating organizations to provide for their own Kafka node to enable consensus in addition to their ordering node(s) instead of relying on other organizations to provide such service. HLF's documentation states that the number of Zookeeper nodes should be either 3, 5, or 7 and that anything over 7 is an overkill [43]. While it can be considered an overkill, it is also in the best interest of the consortium members to manage their own Zookeeper node since any attack that disrupts all Zookeeper nodes (small surface area) would result in complete disruption of the consensus service keeping the HLF network from updating.

5.6.3 Unauthorized Data Access Prevention

There exists the possibility of a participant utilizing the shared ledger data to perform analysis on traffic patterns, origin and destination locations, and the time-of-day specific vehicles are in movement. Results could reveal details such as home address, work location, and daily routines of the owner. For this reason, participating organizations must be transparent in the way they handle and use the data. More importantly, all stakeholders must be aware of the possibility of entities performing analysis on data for purposes other than intended. By utilizing access control policies (ACPs) in HLF, a vehicle owner could prevent an organization, such as a service provider or an insurance company, from obtaining anything more than the minimum required access to the data collected and shared by his or her vehicles.

5.7 Summary

This chapter discussed the findings and results of the scenarios discussed in Chapter 4. Moreover, it outlined observations based on the network and application implementations outlined earlier and solutions to issues encountered during the process. Benefits, drawbacks, and challenges of the proposed and implemented network configuration are presented. It closes with a brief discussion of security and privacy considerations using DLNs.

VI. Conclusion

6.1 Overview

This chapter summarizes the work performed for this research including the design and development of the proposed consortium network. It reiterates contributions of the work and summarizes the observations and analysis of the tested scenarios. Recommendations for HLF and HLC implementations are also discussed. It closes by listing areas of future work which include enhancements to the LTMT, testing the resiliency of the network against attacks, benchmarking the scalability of the solution, and testing efficient algorithms that perform corroboration of reported events.

6.2 Summary

This research is focused on the modeling of collaborative intelligent transportation applications using a distributed ledger network, often referred to as blockchain. It identifies current technology and organizational gaps within the transportation ecosystem that could filled by utilizing DLTs. This work describes the technologies that enable current and future ITS along with security and privacy concerns. It introduces the main concepts behind DLTs and their approach to solve known problems within distributed systems. Current and previous research efforts involving ITSs and DLTs are presented, showcasing solutions to known transportation infrastructure and applications problems.

The use of a consortium DLN deployed and maintained by transportation-related entities is proposed. These participating entities range from businesses to government agencies. A 5-Layer architecture for employing this solution was presented along with a description of each of the layers and associated value to the system as a whole. An analysis of alternatives was performed to evaluate three strong contenders to serve as the framework for implementing the proposed solution. The technologies reviewed were Tendermint, MultiChain, and HLF. These were evaluated based on their capability to function as a platform, availability of API/SDK, membership/permissioning features, ability to execute smart-contracts, and the licensing costs. Based on listed criteria, HLF was selected as the framework for implementing the experimental consortium blockchain network.

A three organization consortium network was designed and implemented using the HLF framework and tools provided by HLC. Each organization has its own peer node, orderer, and membership services provider, similar to what a production implementation for operational use should be. The data models required for implementing the two transportation applications were also listed; these were developed using HLC's modeling language. The process for initiating the network within the virtualization platform was also described.

The LTMT was developed to model transportation-related applications that rely on the execution of distributed code and data from a DLN. It is a lightweight application built with web technologies that enables the modeling and simulation of vehicle and infrastructure behavior and communications that resemble VANET interactions. LTMT allows interaction with a DLN during the execution of a simulation scenario through predefined REST API and WebSocket endpoints, a feature not available in other alternatives. In addition, LTMT's dashboard allows for live tracking of vehicles and RSUs as well as their respective parameters making it trivial to debug solutions both at the application and the DLN.

The two DLN-based applications proposed and developed for deployment to the experimental network aim to reduce the millions of dollars in losses due to fraud every year. The first application described entails the use of the DLN to record odometer readings and help identify odometer tampering fraud. The second application relies on data shared via BSMs in VANETs to create a snapshot of accident events in public roads. The work presented lists the environment assumptions as well as control, independent, and response variables for accomplishing the tasks at hand. A performance evaluation is conducted to characterize the capabilities of the virtualized network within the limited hardware platform it was initiated on. This serves as a benchmark for future developments employing an HLF DLN with high-throughput requirements.

Successful implementation of the distributed network functions, or TPF, and the simulation of the applications aforementioned demonstrate the value of DLTs in future transportation infrastructure solutions. Issues identified throughout the modeling and deployment of the proposed solutions are listed. One example is the use of asset references for applications that receive updates to an asset at a higher rate than it can commit to the blockchain, causing MVCC errors. The use of asset or object IDs in place of object references prevents the chaincode from front loading the referenced assets before the execution of a transaction. This in turn prevents an update that changes the state of the referenced object causing all other transactions that referenced the same object at the same time to be voided.

Load test scenarios with different network block configurations and endorsement policy were conducted to measure the performance of the network with different transaction arrival rates. The results highlighted bottlenecks of the network as implemented in the limited hardware platform, and the limitations in terms of applications that can be executed within it. Load testing also demonstrates an approach for optimizing performance parameters in HLF/HLC network/chaincode implementations that are necessary for high-throughput transportation application requirements. It is important to note that most of the limitations of the network are related to processing transactions that modify the state of the chain; reading from the blockchain can be accomplished without the creation of a transaction by querying the world-state database directly.

One common argument against the use of DLTs is that there are more efficient and better performing technologies for storing and managing large amounts of data such as the highly-scalable distributed database system Cassandra. There are benefits and challenges with both solutions approaches. However, the use of DLTs does more than just provide decentralized execution of transactions and storage of data, it also provides means for decentralized governance. This means that no single administrator has the credentials to modify the state of the network on its own by either attempting to tamper data or make changes to the protocol. Membership in a consortium network means that all protocol modifications are reviewed and accepted by the participants and all changes are logged in the blockchain (e.g., membership, chaincode, create/update/delete). The ability to maintain a log of all transactions in the network along with the identities of the individuals submitting them allows for non-repudiation of actions by all participants of the network, a feature often sought in critical information systems.

In conclusion, the research presented in this thesis demonstrates the feasibility of transportation systems communications utilizing emerging DLTs. However, the type of applications that can reliably utilize this collaborative medium of communication are constrained by the mechanism that ensures the integrity of the data in the blockchain, i.e., consensus. Although the selected technology utilized, HLF, relies on the use of a scalable consensus algorithm, it is still unknown whether this technology could withstand the demands of real-time transportation infrastructure services, and at what scale. Applications such as the ones presented do not need to be immediately committed to the blockchain, are not sensitive to time limitations, and are at risk of being tampered with in conventional systems. These attributes make them suitable uses cases for utilizing such network. There are technologies that have shown promising results with respect to processing and reaching consensus with substantially lower latencies (e.g., Hashgraph's $\sim$ 500K TPS throughput [17]) that could be utilized by HLF to provide fast and BFT consensus.

6.3 Research Contributions

This research has made a number of contributions to the field of transportation infrastructure and communications. The proposed 5-Tier DLN architecture tailored for use in collaborative transportation infrastructure services describes the layers required for a sound DLT implementation within transportation infrastructure. The implementation of the lightweight transportation communications modeling and simulation tool, LTMT, utilized for generating various scenarios during this research, can be applied to other cases such as research into services that use vehicle/RSU data broadcasted in a VANET-like way. The use of HLF and HLC as the framework and tools for implementing the experimental network show their value for producing a distributed data collaboration environment with minimal development and engineering, and without disruption of current services. The approach to optimizing the performance of the network by analyzing the network response metrics through load testing shows how this technology can be optimized to meet different application needs and improve the overall scalability of the network even within a fairly constrained hardware platform. Finally, this research work presented recommendations for modeling the data relationships, described implementation challenges, and potential vulnerabilities of the blockchain network implementation described.

6.4 Future Work

Given the novelty of the work presented and continuous technology developments in both DLT and ITS fields there are many areas that could be further explored, matured, or developed. Listed below are topics of interest that would expand the scope of this research:

- **Resiliency against misbehavior and cyberattacks:** There is little to no research available that characterizes HLF resiliency against misbehavior, man-in-the-middle attacks, or unauthorized relay of data from ordering nodes (potential issue identified in Chapter 4).

- **Efficient corroboration of reported events:** This research presented a simple algorithm for corroborating accident data that was implemented as an HLC TPF. However, time constraints prevented analysis of execution time, memory usage, possible optimizations, and no validations were conducted. Further research and development of optimal corroboration algorithms for identifying fake road reports with high confidence levels may prove of considerable importance for securing transportation infrastructure networks.

- **Benchmarking the performance and scalability of transportation smart-contracts using high-performance infrastructure:** The information systems used for transportation infrastructure are most likely hosted in cloud environments and hardware platforms with substantial computing capabilities. As a result, evaluating the performance of the distributed network while stimulated with realistic scenarios in terms of number of vehicles, RSUs, and participants would help determine whether or not the chosen technology would be appropriate for production deployments.

- **HLF Vehicle Authentication and Communications using VPKI:** This work entails the integration of VPKI schemes into the HLF membership services for authentication and BSM verification. This could allow for the revocation of vehicle certificates as soon as malicious behavior is identified through chaincode operations. Also, it would help develop mechanisms for pseudonym rotation as required by established VPKI standards to operate in conjunction with the DLN.

- **DLT for Transportation Private Data Control:** Since version v1.2, HLF offers private data collections which allow peers to endorse, commit, and query private data on an already established channel [53]. This feature enables the development of blockchains that can provide tighter control of private data (e.g., PII) allowing the overall system to be compliant with regulations such as the European Union's General Data Protection Regulation (GDPR). The scope of the research presented can be expanded to utilize Private Data features on HLF to ensure the privacy of the vehicle owner's identity or give vehicle owners the ability give selected businesses or agencies access to private data in exchange for a service (e.g., insurance, sharing services).

- **LTMT Enhancements and Release:** There are many applications where the developed LTMT can be used for purposes other than the ones presented in this work. Future work could enhance current capabilities provided by the application, improve the simulation scalability, and integrate the timestepped scenario generation to the user interface. Other enhancements can include the development of a graphical wizard for gathering the locations/addresses used for generating a vehicle's driving path and modeling scenarios through the application itself and not its libraries.

6.5 Conclusion

This work demonstrated viable mechanisms to implement a secure, collaborative, and distributed transportation information system for enterprise applications based on DLTs. It highlighted critical characteristics of applications which could benefit from these technologies, including the need for provenance and finality, and an inherent environment of low-trust amongst the participants within an ITS ecosystem. It is clear, however, that DLTs are still in the early stages of development and their second or third order effects on production environments remain unknown. Blockchains and DLTs are not intended to, nor are they able to, solve all problems requiring distributed computing and storage. Accordingly, their implementation should be limited to areas where the aforementioned characteristics are inherent to the operation of the system and the limitations of DLTs do not negatively impact the availability, reliability, and scalability of the ITS application. While such a system will come at a cost, the benefits may well prove to outweigh those costs.

Appendix A. Hyperledger Fabric Crypto Configuration

```
# Copyright IBM Corp. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
# AFIT Transportation Network Cryto Generation Configuration
# Updated By: Luis A. Cintron

# ------------------------------------------------------------------
# "OrdererOrgs" - Definition of organizations managing orderer nodes
# ------------------------------------------------------------------
OrdererOrgs:
  - Name: Orderer
    Domain: orderer.afit.edu
    Specs:
      - Hostname: orderer1
        CommonName: orderer1.afit.edu
        SANS:
          - 192.168.64.21
      - Hostname: orderer2
        CommonName: orderer2.afit.edu
        SANS:
          - 192.168.64.22
      - Hostname: orderer3
        CommonName: orderer3.afit.edu
        SANS:
          - 192.168.64.23


# ------------------------------------------------------------------
# "PeerOrgs" - Definition of organizations managing peer nodes
# ------------------------------------------------------------------
PeerOrgs:
  - Name: Org1
    Domain: org1.afit.edu
    Specs:
      - Hostname: org1.afit.edu
        CommonName: org1.afit.edu
        SANS:
          - "peer0.{{.Domain}}"
          - 192.168.64.21
      - Hostname: peer0.org1.afit.edu
        CommonName: peer0.org1.afit.edu
        SANS:
          - "peer0.{{.Domain}}"
          - 192.168.64.21
    CA:
      Hostname: ca.org1.afit.edu
    Template:
      Count: 2
    Users:
      Count: 1

  - Name: Org2
    Domain: org2.afit.edu
    Specs:
      - Hostname: org2.afit.edu
        CommonName: org2.afit.edu
        SANS:
          - "peer0.{{.Domain}}"
          - 192.168.64.22
      - Hostname: peer0.org2.afit.edu
        CommonName: peer0.org2.afit.edu
        SANS:
          - "peer0.{{.Domain}}"
          - 192.168.64.22
    CA:
      Hostname: ca.org2.afit.edu
    Template:
      Count: 2
    Users:
      Count: 1

  - Name: Org3
    Domain: org3.afit.edu
    Specs:
      - Hostname: org3.afit.edu
        CommonName: org3.afit.edu
        SANS:
          - "peer0.{{.Domain}}"
          - 192.168.64.23
      - Hostname: peer0.org3.afit.edu
        CommonName: peer0.org3.afit.edu
        SANS:
          - "peer0.{{.Domain}}"
          - 192.168.64.23
    CA:
      Hostname: ca.org3.afit.edu
    Template:
      Count: 2
    Users:
      Count: 1
```

Appendix B. Hyperledger Fabric Baseline Network Configuration

```
# Copyright IBM Corp. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
# AFIT Transportation Network Baseline Configuration
# Updated By: Luis A. Cintron
# configtx.yaml

#   Section: Organizations
#   This section defines the different organizational identities which will
#   be referenced later in the configuration.
Organizations:
    - &OrdererOrg
        Name: OrdererMSP
        # ID to load the MSP definition as
        ID: OrdererMSP
        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: crypto-config/ordererOrganizations/orderer.afit.edu/msp
        AdminPrincipal: Role.ADMIN

    - &Org1
        Name: Org1MSP
        ID: Org1MSP
        MSPDir: crypto-config/peerOrganizations/org1.afit.edu/msp
        AnchorPeers:
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
            - Host: 192.168.64.21
              Port: 7051
        AdminPrincipal: Role.ADMIN


    - &Org2
        Name: Org2MSP
        ID: Org2MSP
        MSPDir: crypto-config/peerOrganizations/org2.afit.edu/msp
        AdminPrincipal: Role.ADMIN
        AnchorPeers:
            - Host: 192.168.64.22
              Port: 7051

    - &Org3
        Name: Org3MSP
        ID: Org3MSP
        MSPDir: crypto-config/peerOrganizations/org3.afit.edu/msp
        AdminPrincipal: Role.ADMIN

        AnchorPeers:
            - Host: 192.168.64.23
              Port: 7051

Capabilities:
    # Channel capabilities apply to both the orderers and the peers and must be
    # supported by both.  Set the value of the capability to true to require it.
    Global: &ChannelCapabilities
        # V1.1 for Global is a catchall flag for behavior which has been
        # determined to be desired for all orderers and peers running v1.0.x,
        # but the modification of which would cause incompatibilities.  Users
        # should leave this flag set to true.
        V1_1: true

    # Orderer capabilities apply only to the orderers, and may be safely
    # manipulated without concern for upgrading peers.  Set the value of the
    # capability to true to require it.
    Orderer: &OrdererCapabilities
        # V1.1 for Order is a catchall flag for behavior which has been
        # determined to be desired for all orderers running v1.0.x, but the
        # modification of which  would cause incompatibilities.  Users should
        # leave this flag set to true.
        V1_1: true

    # Application capabilities apply only to the peer network, and may be safely
    # manipulated without concern for upgrading orderers.  Set the value of the
    # capability to true to require it.
    Application: &ApplicationCapabilities
        # V1.2 for Application is a catchall flag for behavior which has been
        # determined to be desired for all peers running v1.0.x, but the
        # modification of which would cause incompatibilities.  Users should
        # leave this flag set to true.
        V1_2: true

#   SECTION: Application
#   This section defines the values to encode into a config transaction or
#   genesis block for application related parameters

Application: &ApplicationDefaults
    # Organizations is the list of orgs which are defined as participants on
    # the application side of the network
    Organizations:
        - *Org1
        - *Org2
        - *Org3

#   SECTION: Orderer
#   This section defines the values to encode into a config transaction or
#   genesis block for orderer related parameters

Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: kafka

    Addresses:
        - orderer1.afit.edu:7050
        - orderer2.afit.edu:7050
        - orderer3.afit.edu:7050

    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 1s


   # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 50

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 99 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        PreferredMaxBytes: 512 KB

    Kafka:
        # Brokers: A list of Kafka brokers to which the orderer connects
        # NOTE: Use IP:port notation
        Brokers:
            - 192.168.64.20:9092
            - 192.168.64.20:10092
            - 192.168.64.20:11092
            - 192.168.64.20:12092

    # Organizations is the list of orgs which are defined as participants on
    # the orderer side of the network
    Organizations:

    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        # BlockValidation specifies what signatures must be included in the block
        # from the orderer for the peer to validate it.
        BlockValidation:
            Type: ImplicitMeta
            Rule: "ANY Writers"

#   Profile
#   Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
Profiles:
    ComposerOrdererGenesis:
        Capabilities:
            <<: *ChannelCapabilities
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Addresses:
                - 192.168.64.21:7050
                - 192.168.64.22:7050
                - 192.168.64.23:7050
            Capabilities:
                <<: *OrdererCapabilities
        Consortiums:
            ComposerConsortium:
                Organizations:
                    - *Org1
                    - *Org2
                    - *Org3
    ComposerChannel:
        Consortium: ComposerConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
                - *Org3
            Capabilities:
                - *ApplicationCapabilities
```

Appendix C. Example Hyperledger Fabric Organization Docker Configuration

```
version: '2'

services:

  # Org1
  ca.org1.afit.edu:
    container_name: ca.org1.afit.edu
    hostname: ca.org1.afit.edu
    image: hyperledger/fabric-ca:1.2.0
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.org1.afit.edu
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=host
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org1.afit.edu-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/ed4471a2f6f90117e095f7e999004ffc7e5b6c99c40ab95091e129aaf8085583_sk -b admin:adminpw -d'
    volumes:
      - ./crypto-config/peerOrganizations/org1.afit.edu/ca/:/etc/hyperledger/fabric-ca-server-config
    extra_hosts:
      - "zookeeperkafka.afit.edu:192.168.64.20"
      - "orderer1.afit.edu:192.168.64.21"
      - "orderer2.afit.edu:192.168.64.22"
      - "orderer3.afit.edu:192.168.64.23"
      - "peer0.org1.afit.edu:192.168.64.21"
      - "peer0.org2.afit.edu:192.168.64.22"
      - "peer0.org3.afit.edu:192.168.64.23"
      - "ca.org1.afit.edu:192.168.64.21"
      - "ca.org2.afit.edu:192.168.64.22"
      - "ca.org3.afit.edu:192.168.64.23"
    network_mode: host

  orderer1.afit.edu:
    container_name: orderer1.afit.edu
    hostname: orderer1.afit.edu
    image: hyperledger/fabric-orderer:1.2.0
    environment:
      - ORDERER_GENERAL_LOGLEVEL=debug
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_GENESISMETHOD=file
      - ORDERER_GENERAL_GENESISFILE=/etc/hyperledger/configtx/composer-genesis.block
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
      - ORDERER_GENERAL_LOCALMSPDIR=/etc/hyperledger/orderer/msp
      - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
      - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
      - ORDERER_KAFKA_VERBOSE=true
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=host
      - CONFIGTX_ORDERER_KAFKA_BROKERS=[zookeeperkafka.afit.edu:9092,zookeeperkafka.afit.edu:10092,zookeeperkafka.afit.edu:11092,zookeeperkafka.afit.edu:12092]
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: orderer
    ports:
      - 7050:7050
    volumes:
      - ./:/etc/hyperledger/configtx
      - ./crypto-config/ordererOrganizations/orderer.afit.edu/orderers/orderer1.afit.edu/msp:/etc/hyperledger/orderer/msp
    extra_hosts:
      - "zookeeperkafka.afit.edu:192.168.64.20"
      - "orderer1.afit.edu:192.168.64.21"
      - "orderer2.afit.edu:192.168.64.22"
      - "orderer3.afit.edu:192.168.64.23"
      - "peer0.org1.afit.edu:192.168.64.21"
      - "peer0.org2.afit.edu:192.168.64.22"
      - "peer0.org3.afit.edu:192.168.64.23"
      - "ca.org1.afit.edu:192.168.64.21"
      - "ca.org2.afit.edu:192.168.64.22"
      - "ca.org3.afit.edu:192.168.64.23"
    network_mode: host

  peer0.org1.afit.edu:
    container_name: peer0.org1.afit.edu
    hostname: peer0.org1.afit.edu
    image: hyperledger/fabric-peer:1.2.0
    environment:
      - CORE_PEER_ID=peer0.org1.afit.edu
      - CORE_PEER_ADDRESS=192.168.64.21:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=192.168.64.21:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.afit.edu:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
      - CORE_LOGGING_LEVEL=debug
      - CORE_CHAINCODE_LOGGING_LEVEL=DEBUG
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=host
      - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/peer/msp
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=192.168.64.21:5984
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_PROFILE_ENABLED=true
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: peer node start
    ports:
        - 7051:7051
        - 7053:7053
    volumes:
      - /var/run/:/host/var/run/
      - ./:/etc/hyperledger/configtx
      - ./crypto-config/peerOrganizations/org1.afit.edu/peers/peer0.org1.afit.edu/msp:/etc/hyperledger/peer/msp
      - ./crypto-config/peerOrganizations/org1.afit.edu/users:/etc/hyperledger/msp/users
    depends_on:
      - orderer1.afit.edu
      - couchdb
    extra_hosts:
      - "zookeeperkafka.afit.edu:192.168.64.20"
      - "orderer1.afit.edu:192.168.64.21"
      - "orderer2.afit.edu:192.168.64.22"
      - "orderer3.afit.edu:192.168.64.23"
      - "peer0.org1.afit.edu:192.168.64.21"
      - "peer0.org2.afit.edu:192.168.64.22"
      - "peer0.org3.afit.edu:192.168.64.23"
      - "ca.org1.afit.edu:192.168.64.21"
      - "ca.org2.afit.edu:192.168.64.22"
      - "ca.org3.afit.edu:192.168.64.23"
    network_mode: host

  couchdb:
    container_name: couchdb
    image: hyperledger/fabric-couchdb:0.4.10
    environment:
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=host
    ports:
      - 5984:5984
    environment:
      DB_URL: http://localhost:5984/member_db
    network_mode: host
```

Appendix D. Hyperledger Fabric Organization VM Start Script

```
#!/bin/bash

#StartFabric.sh
#Starts Org1 VM

# Exit on first error, print all commands.
FABRIC_START_TIMEOUT=15

set -e

Usage() {
        echo ""
        echo "Usage: ./startFabric.sh [-d || --dev]"
        echo ""
        echo "Options:"
        echo -e "\t-d or --dev: (Optional) enable fabric development mode"
        echo ""
        echo "Example: ./startFabric.sh"
        echo ""
        exit 1
}

Parse_Arguments() {
        while [ $# -gt 0 ]; do
                case $1 in
                        --help)
                                HELPINFO=true
                                ;;
                        --np)
                                NOPEER=true
                                ;;
                        --fetch | -f)
                                FETCH=true
                                ;;
            --dev | -d)
                                FABRIC_DEV_MODE=true
                                ;;
                esac
                shift
        done
}

Parse_Arguments $@

if [ "${HELPINFO}" == "true" ]; then
    Usage
fi

# Grab the current directory
DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"
ORG="$(basename "$PWD")"

#if [ "${FABRIC_DEV_MODE}" == "true" ]; then
```

```
#    DOCKER_FILE="${DIR}"/composer/docker-compose-dev-org1.yml
#else
    DOCKER_FILE="${DIR}"/../composer/org1.yml
#fi

docker-compose -f "${DOCKER_FILE}" down
docker-compose -f "${DOCKER_FILE}" up -d

if [ "${NOPEER}" == "true" ]; then
    echo "Done starting up containers..."
        exit
fi

# wait for Hyperledger Fabric to start
# incase of errors when running later commands, issue export FABRIC_START_TIMEOUT=<larger number>
echo "sleeping for ${FABRIC_START_TIMEOUT} seconds to wait for fabric to complete start up"
sleep ${FABRIC_START_TIMEOUT}

BLOCKNAME="composerchannel.block"

if [ "${FETCH}" == "true" ]; then
        # Fetch the channel on Peer 2
        echo -e '\n\n\nFetching Channel on Peer 0:'
        docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.afit.edu/msp" peer0.org1.afit.edu peer channel fetch config -o orderer1.afit.edu:7050 -c composerchannel
        BLOCKNAME="composerchannel_config.block"
else
        # Create the channel on Peer 0
        echo -e '\n\n\nCreating Channel on Peer 0:'
        docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.afit.edu/msp" peer0.org1.afit.edu peer channel create -o orderer1.afit.edu:7050 -c composerchannel -f /etc/hyperledger/configtx/composer-channel.tx
fi
# Join peer0.org1.afit.edu to the channel.
echo -e '\n\n\nJoining Peer 0 to Channel:'
docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.afit.edu/msp" peer0.org1.afit.edu peer channel join -b $BLOCKNAME


if [ "${FABRIC_DEV_MODE}" == "true" ]; then
    echo "Fabric Network started in chaincode development mode"
fi
```

Appendix E. Hyperledger Composer Data Models for Transportation Applications

```
//*
//*AFIT Transportation Network Model Definitions
//*Author: Luis Cintron
//*

namespace org.afit.transportation


//*
//*Participants
//*
abstract participant Member identified by participantId{
  o String participantId
}

participant VehicleOwner extends Member {
  o String Name
  o String driversLicense
}

participant ConsortiumMember extends Member {
  o String Name
  o String Company
}

//*
//*General enumerated types
//*
enum EventType{
  o ACCIDENT
  o DETECTION
  o VEHICLEUPDATE
  o INVALIDODOMETERREADING
}

enum VehicleState{
  o MOVING
  o STOPPED
  o ACCIDENT
  o SPEEDING
  o UNDERSPEEDING
}

enum SensorType {
  o VEHICLE
  o RSU
}

//*
//*Non-asset classes
//*
concept VehicleInfo {
  o VehicleState state
  o Double currentSpeed
  o Double bearing optional
  o Double odometer optional
  o Double rpm optional
  o String vehicleId
}

concept Location {
  o Double lat
  o Double lng
}

concept WitnessedVehicleData {
  o String observedVehicleId
  o String sourceId
  o Location location
  o DateTime eventTimestamp
  o String eventId optional
  o Double speed optional
  o Double heading optional
  o Double distanceFromSource optional
  o String[] behavior optional
  o String[] nearbySensors optional
}


//*
//*Transportation Network Assets
//*
abstract asset Sensor identified by sensorId {
  o String sensorId
  o SensorType sensorType default ="VEHICLE"
}

asset Vehicle extends Sensor {
  o String vin
  o Double odometer default=0.0
  --> VehicleOwner owner optional
  --> RoadEvent[] eventsInvolved optional
}

asset RoadSideUnit extends Sensor {
  o String rsuId
  o Location location
  o String description optional
  --> ConsortiumMember owner optional
}

asset RoadEvent identified by eventId{
  o String eventId
  o Location location
  o DateTime eventTimestamp
  o EventType type
  o String[] vehiclesInvolved
  o WitnessedVehicleData[] witnessedData optional
  o Boolean validated default = false
  --> Sensor source
}

```

```
asset RoadEventWitnessedData identified by id{
  o String id
  o String roadEventId
  o String sourceId
  o WitnessedVehicleData[] witnessedData
}

//*
//*Transactions
//*
abstract transaction VehicleTransaction{
  --> Vehicle vehicle
}

transaction VehicleOdometerUpdateTx extends VehicleTransaction {
  o Double odometer
}

transaction RoadEventTransaction {
  o String eventId
  o Location location
  o DateTime eventTimestamp
  o EventType type
  o String[] vehiclesInvolved
  o WitnessedVehicleData[] witnessedData optional
  o Boolean validated default = false
  o String sourceId
}

transaction EventWitnessDataTransaction {
  o String id
  o String roadEventId
  o String sourceId
  o WitnessedVehicleData[] witnessedData
}


//TX to execute setup of the channel through chaincode
transaction SetupTx {
  o Integer numberOfVehicles
  o Integer numberOfRoadUnits
}


//*
//*Events
//*
event VehicleEvent {
  --> Vehicle vehicle
  o EventType type
  o String referenceTxId optional
}
```

Appendix F. Hyperledger Composer Chaincode Logic for Transportation Applications

```
/**
 * AFIT Transportation Network
 * Chaincode
 * Author: Luis Cintron
 */

'use strict';

/**
 * Transction processor functions (Chaincode)
 */

/**
 * Odometer Update transaction
 * @param {org.afit.transportation.VehicleOdometerUpdateTx} tx
 * @transaction
 */
async function vehicleOdometerUpdate(tx) {
  // Save the old value of the asset.
  const odometer = tx.vehicle.odometer;

  // Update the asset with the new value.
  if (tx.odometer > odometer) {
    tx.vehicle.odometer = tx.odometer;

    // Get the asset registry for the asset.
    const assetRegistry = await getAssetRegistry('org.afit.transportation.Vehicle');

    // Update the asset in the asset registry.
    await assetRegistry.update(tx.vehicle);

    // Emit an event for the modified asset.
    const factory = getFactory();
    let event = factory.newEvent('org.afit.transportation', 'VehicleEvent');
    event.vehicle = tx.vehicle;
    event.type = 'VEHICLEUPDATE'
    event.referenceTxId = tx.transactionId
    emit(event);
  } else {
    // Emit an event for the invalid odometer reading attempt.
    const factory = getFactory();
    let event = factory.newEvent('org.afit.transportation', 'VehicleEvent');
    event.vehicle = tx.vehicle;
    event.type = 'INVALIDODOMETERREADING'
    event.referenceTxId = tx.transactionId
    emit(event);
  }
}

/**
 * Network data initialization - Create owners/members & ids
 * @param {org.afit.transportation.SetupTx} tx
 * @transaction
 */
async function setup(tx) {
  const factory = getFactory();
  const namespace = "org.afit.transportation";

  if (tx.numberOfVehicles) {
    const rreg = await getAssetRegistry(namespace + '.Vehicle');
    for (i = 0; i < tx.numberOfVehicles; i++) {
      let v = factory.newResource(namespace, 'Vehicle', 'V' + i);
      v.vin = 'vin';
      v.odometer = 0.0;
      v.eventsInvolved = [];
      v.sensorId = 'V' + i;
      await rreg.add(v);
    }
  }

  if (tx.numberOfRoadUnits) {
    const rreg = await getAssetRegistry(namespace + '.RoadSideUnit');
    for (i = 0; i < tx.numberOfRoadUnits; i++) {
      let r = factory.newResource(namespace, 'RoadSideUnit', 'R' + i);
      r.rsuId = 'R' + i;
      r.location = factory.newConcept(namespace,'Location');
      r.location.lat=0.0;
      r.location.lng=0.0;
      r.sensorId = 'R' + i;
      r.sensorType = 'RSU';
      await rreg.add(r);
    }
  }
}


/**
 * Event Report Transaction
 * @param {org.afit.transportation.RoadEventTransaction} tx
 * @transaction
 */
async function RoadEventReport(tx) {
  const factory = getFactory();
  const namespace = "org.afit.transportation";

  const re = factory.newResource(namespace, 'RoadEvent', tx.eventId);
  re.location = tx.location
  re.type = tx.type;
  re.eventTimestamp = tx.eventTimestamp;
  re.vehiclesInvolved = tx.vehiclesInvolved;
  re.witnessedData = tx.witnessedData;
  re.validated = false;

  if (tx.sourceId.startsWith('R')) {
    const rreg = await getAssetRegistry(namespace + '.RoadSideUnit');
    re.source = await rreg.get(tx.sourceId);
  }
  else {
    const vreg = await getAssetRegistry(namespace + '.Vehicle');
    re.source = await vreg.get(tx.sourceId);
  }

  const rereg = await getAssetRegistry(re.getFullyQualifiedType());
  let result = await rereg.add(re);

  //updated vehicles involved
  const vreg = tx.vehiclesInvolved.length ? await getAssetRegistry(namespace + '.Vehicle') : null;
  for (vid of tx.vehiclesInvolved) {
    const vehicle = await vreg.get(vid);
    if (!vehicle.eventsInvolved)
      vehicle.eventsInvolved = [];

    vehicle.eventsInvolved.push(re);
    await vreg.update(vehicle);
  }

}


/**
 * Event Report Transaction
 * @param {org.afit.transportation.EventWitnessDataTransaction} tx
 * @transaction
 */
async function EventWitnessedDataReport(tx) {
  const factory = getFactory();
  const namespace = "org.afit.transportation";
  if (tx.witnessedData.length > 0) {
    const rereg = await getAssetRegistry(namespace + '.RoadEvent');
    const reExists = await rereg.exists(tx.roadEventId);
    if(reExists){
      const rwd = factory.newResource(namespace, 'RoadEventWitnessedData', tx.id);
      rwd.roadEventId = tx.roadEventId;
      rwd.witnessedData=tx.witnessedData;
      rwd.sourceId = tx.sourceId;
      const rwdReg = await getAssetRegistry(rwd.getFullyQualifiedType());
      let result = await rwdReg.add(rwd);
    }
  }
}
```

Appendix G. AutoComposer - Script for Generating and Deploying Hyperledger Composer Network Archives

```bash
#!/bin/bash

# Hyperledger Composer Custom Business Network Commands
# Author: Luis Cintron
# ------------------------------------------------------------------------
# HELP:
#   -Create business network archive (requires version or version-upgrade):
#       $ ./autoComposer -c -v 0.0.1  or  $ ./autoComposer -c -v 0.0.1-0.0.2 (update ver to 0.0.2)
#
#   -Install business network:
#       $ ./autoComposer -x -v 0.0.1
#
#   -Start business network:
#       $ ./autoComposer -s -v 0.0.1
# ------------------------------------------------------------------------


ORG="$(hostname)"
#Global variables
CREATE=false
START=false
UPGRADE=false
PING=false
REST=false
WEB=false
IMPORT=false
ANGULAR=false
VERSIONSET=false
INSTALL=false
HELP=false
#Grab passed arguments and set global variables
while getopts :hacxsuiprwv: option
do
case "${option}"
in
h) HELP=true;;
a) ANGULAR=true;;
c) CREATE=true;;
x) INSTALL=true;;
s) START=true;;
u) UPGRADE=true;;
i) IMPORT=true;;
p) PING=true;;
r) REST=true;;
w) WEB=true;;
v) VERSIONSET=true
   VERSION=${OPTARG};;
esac
```

```
done

#functions
execute_composer(){
    if [[ $($@ | grep succeeded)  == *"succeeded"* ]]; then
        return 0
    else
        echo "Error executing '$@'"
        #exit
        return 1
    fi
}

if [ "$HELP" = true ]; then
    echo "autoComposer - Hyperledger Composer automation"
    echo -e "Usage: autoComposer.sh <command>\n"
    echo    "Commands:"
    echo -e "\t-c -v <version>\tCreate a Business network archive of version provided"
    echo -e "\t-c -v <oldversion>-<newversion>\tCreate a Business network archive and updates old version to new version"
    echo -e "\t-x -v <version>\tInstall a Business network archive of version provided"
    echo -e "\t-s -v <version>\tStart a Business network of version provided"
    echo -e "\t-u -v <version>\tUpgrade a Business network to version provided"
    echo -e "\n\t-i Import org admin credentials - based on hostname - results in orgadmin@afit-transportation..."
    echo -e "\t-p Ping the business network using the orgadmin card"
    echo -e "\t-r Start composers rest server"
    echo -e "\t-a Generates an Angular Web Application from Template (Baseline)"
    echo -e "\t-w Runs generated Angular Web Application\n"

    exit
fi

#Create network. This will prompt for network name, license, etc.
#yo hyperledger-composer:businessnetwork

if ([ "$CREATE" = true ] || [ "$INSTALL" = true ] || [ "$START" = true ] || [ "$UPGRADE" = true ] ) && [ "$VERSIONSET" = false ]; then
    echo "Missing version (-v) for creating/starting/upgrading the network"
    exit
fi

#Once structure has been modified/generated (models, logic, rules, queries, etc.)
if [ "$CREATE" = true ] && [ "$VERSIONSET" = true ]; then

    #Update Composer network old version to new version in package.json
    if [[ $VERSION == *-* ]]; then
        LAST_VERSION=$(echo $VERSION | cut -d'-' -f1) # output is 1
        VERSION=$(echo $VERSION | cut -d'-' -f2) # output is 2
        echo "Parse Versions: $LAST_VERSION to $VERSION"
    fi

    if [ "$VERSIONSET" = true ] && [ "$VERSION" != "$LAST_VERSION" ]; then
        CHECK="s/\\\"version\\\": \\\"$LAST_VERSION\\\"/\\\"version\\\": \\\"$VERSION\\\"/g"
        echo "Replacing version (Using: $CHECK)"
        sed -i -e "$CHECK" package.json
    fi

    echo "Creating archive - Version $VERSION..."
    #Generate the business network archive
    if execute_composer "composer archive create -t dir -n ."; then
        echo "Archive ver. $VERSION created!"
        echo ""
    fi
fi

if [ "$INSTALL" = true ] && [ "$VERSIONSET" = true ]; then
    #Install business network
    if execute_composer "composer network install --card PeerAdmin@afittransnetwork --archiveFile afit-transportation-network@$VERSION.bna"; then
        echo "Network ver. $VERSION installed!"
        echo ""
    else
        echo "[ERROR] Unable to install ver. $VERSION. Exiting..."
        echo ""
        exit
    fi
fi

#Start business network
if [ "$START" = true ] && [ "$VERSIONSET" = true ]; then
    echo "Starting the network..."
    if [ ! -d "${ORG}/admin" ]; then
        mkdir -p "${ORG}/admin"
    fi
    composer network start --networkName afit-transportation-network --networkVersion $VERSION -A org1admin -C org1/admin/admin-pub.pem  -A org2admin -C org2/admin/admin-pub.pem -A org3admin -C org3/admin/admin-pub.pem --card PeerAdmin@afittransnetwork
    echo "Network ver. $VERSION started!"
    echo ""
fi

#Upgrade business network
if [ "$UPGRADE" = true ] && [ "$VERSIONSET" = true ]; then
    echo "Upgrading the network..."
    if execute_composer "composer network upgrade -c
    PeerAdmin@afittransnetwork -n afit-transportation-network -V $VERSION"; then
        echo "Network upgraded to ver. $VERSION"
        echo ""
    else
        echo "[ERROR] Unable to upgrade to ver. $VERSION. Exiting..."
        echo ""
        exit
    fi
fi

#Import network admin card
if [ "$IMPORT" = true ]; then
    NETCARD="$(composer card list -q | grep afit-transportation-network)"
    if [ "${NETCARD}" != "${ORG}admin@afit-transportation-network" ]; then
        echo "Card has not been created. Creating..."
        composer identity request -c PeerAdmin@afittransnetwork -u "${ORG}admin" -s adminpw -d "${ORG}/admin"
        composer card create -p "${ORG}/connection.json" -u "${ORG}admin" -n afit-transportation-network -c "${ORG}/admin/admin-pub.pem" -k "${ORG}/admin/admin-priv.pem" -f "${ORG}/networkAdmin.card" -r PeerAdmin -r ChannelAdmin
    fi

    echo "Importing network admin card..."
    execute_composer "composer card delete -c ${ORG}admin@afit-transportation-network"
    if execute_composer "composer card import --file ${ORG}/networkAdmin.card"; then
        echo "${ORG} Networkadmin.card imported!"
        echo ""
    else
        echo "[ERROR] Unable to import network admin card. Exiting..."
        echo ""
        exit
    fi
fi

#Check network deployment succeeded
if [ "$PING" = true ]; then
    echo "Pinging the network..."
    if execute_composer "composer network ping --card ${ORG}admin@afit-transportation-network"; then
        echo "Network is up and running!"
    else
        echo "[ERROR] The network did not respond!"
        exit 1
    fi
    echo ’’
fi

#Generate & Start Rest Server
```

```
if [ "$REST" = true ]; then
    echo "Starting composer-rest-server..."
    kill $(lsof -i :3000 | grep -i :3000 | awk '{print $2}') 2>&1
    composer-rest-server -c "${ORG}admin@afit-transportation-network" -n never -u true -w true
fi


#Generate Angular application
if [ "$ANGULAR" = true ]; then
    if [ -d "./afit-transportation-webapp" ]; then
        echo "Stopping angular app..."
        cd ./afit-transportation-webapp
        npm stop 2>&1
        sleep 2
        cd ..
        rm -r -f ./afit-transportation-webapp 2>&1
    fi

    echo "Generating the angular shell..."
    yo hyperledger-composer:angular
fi

#Start Angular application
if [ "$WEB" = true ]; then
    echo "Starting angular app..."
    cd ./afit-transportation-webapp/
    kill $(lsof -i :4200 | grep -i :4200 | awk '{print $2}')
    npm start 2&>1
fi

echo 'Done.'
```

Appendix H. Hyperledger Composer Connection Profile Example (connection.json)

```
{
    "name": "afittransnetwork",
    "x-type": "hlfv1",
    "x-commitTimeout": 300,
    "version": "1.0.0",
    "client": {
        "organization": "Org2",
        "connection": {
            "timeout": {
                "peer": {
                    "endorser": "300",
                    "eventHub": "300",
                    "eventReg": "300"
            },
                "orderer": "300"
            }
        }
    },
    "channels": {
        "composerchannel": {
            "orderers": [
                "orderer1.afit.edu",
                "orderer2.afit.edu",
                "orderer3.afit.edu"
            ],
            "peers": {
                "peer0.org1.afit.edu": {
                    "endorsingPeer": false,
                    "chaincodeQuery": true,
                    "ledgerQuery":true,
                    "eventSource":true
                },
                "peer0.org2.afit.edu": {
                    "endorsingPeer": true,
                    "chaincodeQuery": true,
                    "ledgerQuery":true,
                    "eventSource":true
                },
                "peer0.org3.afit.edu": {
                    "endorsingPeer": true,
                    "chaincodeQuery": true,
                    "ledgerQuery":true,
                    "eventSource":true
                }
            }
        }
    },
    "organizations": {
        "Org1": {
            "mspid": "Org1MSP",
            "peers": [
                "peer0.org1.afit.edu"
            ],
            "certificateAuthorities": [
                "ca.org1.afit.edu"
            ]
        },
        "Org2": {
            "mspid": "Org2MSP",
            "peers": [
                "peer0.org2.afit.edu"
            ],
            "certificateAuthorities": [
                "ca.org2.afit.edu"
            ]
        },
        "Org3": {
            "mspid": "Org3MSP",
            "peers": [
                "peer0.org3.afit.edu"
            ],
            "certificateAuthorities": [
                "ca.org3.afit.edu"
            ]
        }
    },
    "orderers": {
        "orderer1.afit.edu": {
            "url": "grpc://192.168.64.21:7050"
        },
        "orderer2.afit.edu": {
            "url": "grpc://192.168.64.22:7050"
        },
        "orderer3.afit.edu": {
            "url": "grpc://192.168.64.23:7050"
        }
    },
    "peers": {
        "peer0.org1.afit.edu": {
            "url": "grpc://192.168.64.21:7051"
        },
        "peer0.org2.afit.edu": {
            "url": "grpc://192.168.64.22:7051"
        },
        "peer0.org3.afit.edu": {
            "url": "grpc://192.168.64.23:7051"
        }
    },
    "certificateAuthorities": {
        "ca.org1.afit.edu": {
            "url": "http://192.168.64.21:7054",
            "caName": "ca.org1.afit.edu"
        },
        "ca.org2.afit.edu": {
```

```
            "url": "http://192.168.64.22:7054",
            "caName": "ca.org2.afit.edu"
        },
        "ca.org3.afit.edu": {
            "url": "http://192.168.64.23:7054",
            "caName": "ca.org3.afit.edu"
        }

    }
}
```

Appendix I. Performance Test Matrix

Arrival-Rate (TPS)	Endorsement Policy	Block Configuration
10	All Orgs	10 TPB - 2 s
20	All Orgs	10 TPB - 2 s
30	All Orgs	10 TPB - 2 s
40	All Orgs	10 TPB - 2 s
50	All Orgs	10 TPB - 2 s
60	All Orgs	10 TPB - 2 s
70	All Orgs	10 TPB - 2 s
80	All Orgs	10 TPB - 2 s
90	All Orgs	10 TPB - 2 s
100	All Orgs	10 TPB - 2 s
10	2-Of	10 TPB - 1 s
20	2-Of	10 TPB - 1 s
30	2-Of	10 TPB - 1 s
40	2-Of	10 TPB - 1 s
50	2-Of	10 TPB - 1 s
60	2-Of	10 TPB - 1 s
70	2-Of	10 TPB - 1 s
80	2-Of	10 TPB - 1 s
90	2-Of	10 TPB - 1 s
100	2-Of	10 TPB - 1 s
10	All Orgs	50 TPB - 1 s
20	All Orgs	50 TPB - 1 s
30	All Orgs	50 TPB - 1 s
40	All Orgs	50 TPB - 1 s

50	All Orgs	50 TPB - 1 s
60	All Orgs	50 TPB - 1 s
70	All Orgs	50 TPB - 1 s
80	All Orgs	50 TPB - 1 s
90	All Orgs	50 TPB - 1 s
100	All Orgs	50 TPB - 1 s
10	2-Of	50 TPB - 1 s
20	2-Of	50 TPB - 1 s
30	2-Of	50 TPB - 1 s
40	2-Of	50 TPB - 1 s
50	2-Of	50 TPB - 1 s
60	2-Of	50 TPB - 1 s
70	2-Of	50 TPB - 1 s
80	2-Of	50 TPB - 1 s
90	2-Of	50 TPB - 1 s
100	2-Of	50 TPB - 1 s
10	All Orgs	100 TPB - 1 s
20	All Orgs	100 TPB - 1 s
30	All Orgs	100 TPB - 1 s
40	All Orgs	100 TPB - 1 s
50	All Orgs	100 TPB - 1 s
60	All Orgs	100 TPB - 1 s
70	All Orgs	100 TPB - 1 s
80	All Orgs	100 TPB - 1 s
90	All Orgs	100 TPB - 1 s
100	All Orgs	100 TPB - 1 s
10	2-Of	100 TPB - 1 s
20	2-Of	100 TPB - 1 s

30	2-Of	100 TPB - 1 s
40	2-Of	100 TPB - 1 s
50	2-Of	100 TPB - 1 s
60	2-Of	100 TPB - 1 s
70	2-Of	100 TPB - 1 s
80	2-Of	100 TPB - 1 s
90	2-Of	100 TPB - 1 s
100	2-Of	100 TPB - 1 s

Appendix J. Performance Test Workflow Scripts

multiworkflow.sh

```
#!/bin/bash
START=0
END=0
INCREMENT=0
REPEAT=1
WAIT=0
DATESTR=$(date +"%Y-%m-%d_%H-%M-%S-%1N")
OUTPUTFILE="../../Data/$DATESTR-MULTI"
while getopts :s:e:i:r:w: option
do
case "${option}"
in
s) START=${OPTARG};;
e) END=${OPTARG};;
i) INCREMENT=${OPTARG};;
r) REPEAT=${OPTARG};;
w) WAIT=${OPTARG};;
esac
done
echo -e "Scenario Collection:\t $DATESTR-MULTI.txt"
for i in ‘seq $START $INCREMENT $END‘
do
    for j in ‘seq 1 1 $REPEAT‘
    do
        echo "Calling workflow -r $i"

        ./workflow.sh -r $i >> "$OUTPUTFILE.txt"
        echo "---" >> "$OUTPUTFILE.txt"
        sleep $WAIT
    done
done
echo "Generating $DATESTR-MULTI.csv..."
python ./TextToCsv.py -i "$OUTPUTFILE.txt" > "$OUTPUTFILE.csv"

echo "Job finished!"
```

workflow.sh

```bash
#!/bin/bash
FETCHBLOCK=true
SLEEPTIME=20
EVENTPREFIXSET=false
REQUESTSET=false
REQUEST=500
QUIET=false
BLOCKNUM=6
TEST=false
IN=0
WAITT=10
DATESTR=$(date +"%Y-%m-%d_%H-%M-%S-%1N")

while getopts :r:b:e:s:q:t:w: option
do
case "${option}"
in
b) BLOCKNUM=${OPTARG}
   FETCHBLOCK=false;;
r) REQUEST=${OPTARG};;
t) TEST=true
   IN=${OPTARG};;
w) WAITT=${OPTARG};;
esac
done

if [ "$EVENTPREFIXSET" = false ]; then
    EVENTPREFIX=$DATESTR
fi

if [ "$FETCHBLOCK" = true ]; then
    BLOCKNUM=$(node ./GetNextBlockNumber.js)
fi

if [ "$TEST" = true ]; then
    node TestBounds.js $DATESTR $REQUEST $IN $WAITT
else
    node GenerateEvents.js $REQUEST $DATESTR $BLOCKNUM
fi

#Wait for Explorer to Update
sleep $SLEEPTIME

node GetBlockTxDataToCSV.js $DATESTR $BLOCKNUM $REQUEST
```

GenerateEvents.js

```
var blockNum = 0;
var request = require('request');
var reqProctimes = [];
var args = process.argv.slice(2);
var numEvents = 1;
var errorTotal = 0;
var successTotal = 0;
var timeoutErr =0;
var hlfError = 0;
var count = 0;
var rendTime ='';
var requestEndTime = '';
var scenarioEndTime = {};
var scenarioName = "";
var endpoints = [
    "http://192.168.64.21:3000/api/RoadEventTransaction",
    "http://192.168.64.22:3000/api/RoadEventTransaction",
    "http://192.168.64.23:3000/api/RoadEventTransaction"
]

//Initialize variables based on passed CLI parameters
if(args.length>0){
    numEvents = parseInt(args[0]);
    scenarioName = args[1];
    blockNum = parseInt(args[2]);
}

console.log("Scenario:\t\t"+ scenarioName);
console.log("Events:\t\t\t"+ numEvents);
console.log("Start Block:\t\t"+ blockNum);

//Allows find-replace for more than one appearances
//of the substring in the string to search
String.prototype.replaceAll = function(search, replacement) {
    var target = this;
    return target.replace(new RegExp(search, 'g'), replacement
       );
};

//Makes an HTTP POST Request to a given url,
//with given data. Finds the last request response time.
function HTTPPostRequest(request_url,data, success_callback,
   error_callback, index = 0) {
    reqProctimes[index].startTime= new Date();
    request({url:request_url,method:"POST", json: data,
       headers: {
        "content-type": "application/json",
        },}, function (err, res, json) {
        if (err && error_callback) {
            error_callback(err);
            errorTotal++;
            timeoutErr++;
        }
        else if(json && json.eventId) {
            if(success_callback)
```

```
                success_callback(json, index);
            successTotal++;
        }
        else{
            errorTotal++
            hlfError++;
        }
        scenarioEndTime = new Date();
        reqProctimes[index].endTime= new Date();
        count--;
    });
    reqCount++;
    if(reqCount == numEvents){
        rendTime = new Date();
        requestEndTime = (rendTime.getTime() - startTime.getTime())/1000;
    }
};

//Submit RoadEvent transactions for load testingt
var reqCount = 0;
var SOURCEVEHICLE = 0;
var timePerReq = 1000/numEvents;
var timeCount = 0;
var startTime = new Date();
for(i = 0; i < numEvents;i++){
    var endpoint = endpoints[i%3];
    OTHERVEHICLE = SOURCEVEHICLE+1;
    let data = {};
    data.$class="org.afit.transportation.RoadEventTransaction"
    data.eventId=scenarioName+'-'+i;
    data.location ={};
    data.location.$class="org.afit.transportation.Location";
    data.location.lat =158.814
    data.location.lng =  224.262;
    data.eventTimestamp = new Date().toISOString();
    data.type = "ACCIDENT";
    data.vehiclesInvolved=["V"+SOURCEVEHICLE,"V"+OTHERVEHICLE];
    data.validated= false;
    data.sourceId= "V"+SOURCEVEHICLE;

    count++;
    reqProctimes.push({startTime:'', endtime:''});
    setTimeout(HTTPPostRequest, timeCount+(i*timePerReq), endpoint, data, null, null,i);
    SOURCEVEHICLE = SOURCEVEHICLE+2;
}


//Wait for all HTTP POST requests to return and calculates errors,
//mean request time, total time elapsed, etc.
var interval = setInterval(function () {
    if (count == 0) {
        var meanReqProcTime = 0;
        var totalReqProcTime = 0;
        for(i = 0;i<numEvents;i++){
            reqProctimes[i].reqTime =(reqProctimes[i].endTime.getTime()-reqProctimes[i].startTime.getTime())/1000;
            totalReqProcTime+=reqProctimes[i].reqTime;
        }
        var meanReqProcTime = totalReqProcTime/reqProctimes.length;
        clearInterval(interval);
        scenarioEndTime = (scenarioEndTime.getTime() - startTime.getTime())/1000;

        console.log("Request Submit Time: \t"+requestEndTime);
        console.log("Request/sec:\t\t"+(numEvents/requestEndTime));
        console.log("Mean Req Time:\t\t"+meanReqProcTime);
        console.log("Total Time Elapsed:\t"+scenarioEndTime);
        console.log("Successful Requests:\t"+successTotal);
        console.log("Timeout Errors:\t\t"+timeoutErr);
        console.log('HLF Errors:\t\t'+hlfError);
        console.log("Total Error:\t\t"+errorTotal);
    }
}, 2000);
```

TestBounds.js

```
var request = require(’request’);
var args = process.argv.slice(2);
var errorTotal = 0;
var successTotal = 0;
var EVENTPREFIX = new Date().getTime() / 1000;
var scenarioEndTime = {};
var scenarioName = "";
var numEvents = 1;
var increments = 10;
var waitTime = 2;
var endpoints = [
    "http://192.168.64.21:3000/api/RoadEventTransaction",
    "http://192.168.64.22:3000/api/RoadEventTransaction",
    "http://192.168.64.23:3000/api/RoadEventTransaction"
]
if(args.length>0){
    scenarioName = args[0]
    numEvents = parseInt(args[1]);
    increments = parseInt(args[2]);
    waitTime = parseInt(args[3]);

}

console.log("Sending " + numEvents + " events in increments of "+increments+" every "+waitTime+"ms...")

String.prototype.replaceAll = function(search, replacement) {
    var target = this;
    return target.replace(new RegExp(search, ’g’), replacement);
};

function HTTPPostRequest(request_url,data, success_callback, error_callback, isJSONRequest, index = 0) {
    request({url:request_url,method:"POST", json: data, headers: {
        "content-type": "application/json",
        },}, function (err, res, json) {
        if (err && error_callback) {
            error_callback(err);
            errorTotal++;
        }
        else if(json.eventId) {
            if(success_callback)
                success_callback(json, index);
            successTotal++;
        }
        else{
            console.log(json);
            errorTotal++
        }
        scenarioEndTime = new Date();
        count--;

    });

};

//Get all BlockTx info
var SOURCEVEHICLE = 0;
var timeoutSum =0;
var count = numEvents;
var eventId = 0;
var reqs = increments;

function sendRequests(x){
    console.log("Sending " + x + " events.");
    for(i = 0; i < x;i++){
        var endpoint = endpoints[i%3];
        OTHERVEHICLE = SOURCEVEHICLE+1;
        let data = {};
        data.$class="org.afit.transportation.RoadEventTransaction"
        data.eventId=scenarioName+'-'+eventId++;
        data.location ={};
        data.location.$class="org.afit.transportation.Location";
        data.location.lat =158.814
        data.location.lng =  224.262;
        data.eventTimestamp = new Date().toISOString();
        data.type = "ACCIDENT";
        data.vehiclesInvolved=["V"+SOURCEVEHICLE%3000,"V"+OTHERVEHICLE%3000];
        data.validated= false;
        data.sourceId= "V"+SOURCEVEHICLE%3000;
        HTTPPostRequest(endpoint, data);
        SOURCEVEHICLE = SOURCEVEHICLE+2;
    }
}
while(reqs<=numEvents){

    setTimeout(sendRequests, timeoutSum, reqs);
    reqs = reqs + increments;
    timeoutSum+=waitTime;
}
```

GetBlockTxData.js

```
const request = require(’request’);
const uri = ’http://192.168.64.21:8081/api/’;

function GetBlockTxListUri(channel_genesis){
    return uri+’blockAndTxList/’+channel_genesis+’/0’;
}

function GetTransactionUri(channel_genesis, txid){
    return uri+’transaction/’+channel_genesis+’/’+txid;
}

function HTTPGetRequest(request_url, success_callback, error_callback, isJSONRequest, index=0) {
    request({ url: request_url, json: isJSONRequest }, function (err, res, json) {
        if (err && error_callback) {
            error_callback(err);
        }
        else {
            success_callback(json,index);
        }
    });
};

var blocks = [];
let gb = ’’;
var count = 0;
//Get all Block TX Info
count++;
request(GetBlockTxListUri(gb), { json: true }, (err, res, body) => {
  if (err) { return console.log(err); }
  if(body.rows){
        blocks.push(...body.rows);
        blocks.forEach(block =>{
            block.transactions = [];
            //For each block transaction get details
            block.txhash.forEach(tx =>{
                count++;
                request(GetTransactionUri(gb,tx),{ json: true }, (err, res, body) =>{
                    if(err) return console.log(err);
                    if(body.row && body.row.chaincodename === ’afit-transportation-network’){
                        var txTime = new Date(body.row.createdt);
                        var bTime= new Date(block.createdt);
                        var transaction = {};
                        transaction.hash = body.row.hash;
                        transaction.createdt = body.row.createdt;
                        transaction.write_set = body.row.write_set;
                        transaction.endorsers = body.row.endorser_msp_id;
                        transaction.creator = body.row.creator_msp_id;
                        transaction.timeToBlock = bTime - txTime;
                        transaction.validation_code = body.row.validation_code;
                        block.transactions.push(transaction);
                    }
                    count--;
                });
            });
        });
  }
  count--;
});

//Wait to fetch all TX data then output all blocks and TX data
//to console.
var interval = setInterval(function(){
    if(count == 0){
        console.log(JSON.stringify(blocks));
        clearInterval(interval);
    }
},100);
```

Appendix K. Performance Scenario Sample Data Output

Scenaric	Req No.	Block	Req Submit Time	Req/s	Avg Total Req Time	Total Time Elapsed	Succe	Timeout Error	HLF Error	Total Errors	Avg TPS	Peak TPS	Total Block	Avg BPS	Peak BPS	Avg TPB	Peak TPB	Min TPB	Total Tx	Total Time
1	10	6	0.902	11.086	1.1162	1.576	10	0	0	0	5	6	1	1	1	10	10	10	10	1
2	10	7	0.903	11.074	0.901	1.36	10	0	0	0	5	7	1	1	1	10	10	10	10	1
3	10	8	0.902	11.086	0.9405	1.395	10	0	0	0	5	7	1	1	1	10	10	10	10	1
4	20	9	0.951	21.030	1.56615	2.505	20	0	0	0	10	14	2	1	1	10	10	10	20	1
5	20	11	0.951	21.030	1.4166	2.532	20	0	0	0	10	16	2	1	1	10	15	5	20	1
6	20	13	0.951	21.030	1.66775	2.654	20	0	0	0	10	17	2	1	1	10	11	9	20	1
7	30	15	0.968	30.991	2.2277	2.99	30	0	0	0	15	28	2	2	2	15	20	10	30	1
8	30	17	0.968	30.991	2.139566667	2.935	30	0	0	0	15	22	2	2	2	15	19	11	30	1
9	30	19	0.968	30.991	2.1227	2.721	30	0	0	0	15	16	2	2	2	15	25	5	30	1
10	40	21	0.976	40.983	2.67125	3.245	40	0	0	0	20	33	2	2	2	20	35	5	40	1
11	40	23	0.977	40.941	2.978225	3.731	40	0	0	0	20	31	2	2	2	20	27	13	40	1
12	40	25	0.978	40.899	2.962325	3.612	40	0	0	0	20	28	2	2	2	20	32	8	40	1
13	50	27	0.981	50.968	3.344	3.894	50	0	0	0	25	34	2	2	2	25	47	3	50	1
14	50	29	0.983	50.864	3.4654	4.104	50	0	0	0	25	49	2	1	1	25	43	7	50	1
15	50	31	0.982	50.916	3.3642	3.903	50	0	0	0	25	39	2	2	2	25	47	3	50	1
16	60	33	0.986	60.851	3.927666667	4.751	60	0	0	0	30	39	2	1	1	30	40	20	60	1
17	60	35	0.987	60.790	4.76945	5.74	60	0	0	0	30	38	3	1.5	2	20	45	4	60	1
18	60	38	0.986	60.851	3.830566667	4.782	60	0	0	0	30	50	2	1	1	30	30	30	60	1
19	70	40	0.988	70.850	4.556257143	5.589	70	0	0	0	35	67	3	3	3	23.33	50	10	70	1
20	70	43	0.988	70.850	4.508871429	5.546	70	0	0	0	35	39	3	1.5	2	23.33	50	5	70	1
21	70	46	0.987	70.921	4.343228571	5.356	70	0	0	0	35	64	2	1	1	35	47	23	70	1
22	80	48	0.989	80.889	5.139575	6.198	80	0	0	0	40	52	3	1.5	2	26.66	50	13	80	1
23	80	51	0.989	80.889	5.1492	6.122	80	0	0	0	40	52	3	1.5	2	26.66	50	11	80	1
24	80	54	0.991	80.726	4.9079125	5.908	80	0	0	0	40	51	3	1.5	2	26.66	50	6	80	1
25	90	57	0.99	90.909	5.538566667	6.577	90	0	0	0	45	73	3	1.5	2	30	50	19	90	1
26	90	60	0.99	90.909	5.797511111	6.79	90	0	0	0	45	62	3	1.5	2	30	50	3	90	1
27	90	63	0.991	90.817	5.714155556	6.78	90	0	0	0	45	72	3	1.5	2	30	50	11	90	1
28	100	66	0.991	100.90	6.44078	7.508	100	0	0	0	50	54	3	1.5	2	33.33	50	22	100	1
29	100	69	0.992	100.80	6.55511	7.644	100	0	0	0	50	55	3	1.5	2	33.33	50	13	100	1

Appendix L. Julia Analysis Script

```julia
using DataFrames, CSV, Statistics, Gadfly, Cairo, Suppressor
using Genie
import Genie.Router: route
import Genie.Renderer: json!

#Fetch scenario data from CSV file in local directory and
#perform calculations on data. Return results in the web response
function AnalyzeBCData()
    html_text=""

    #Find latest generated .csv data file in Data directory
    wd = pwd()
    dataPath="../../Data/"
    cd(dataPath)
    rd = filter(x ->occursin(".csv",x), readdir(pwd()))
    val,ind = findmax(map(ctime,rd))
    fileName = split(rd[ind],".csv")[1]
    filePath= "../../Data/"*fileName
    cd(wd)

    #Read sceanario .csv data
    df = CSV.read(filePath*".csv", delim=",")

    #Sort by block number
    df = sort!(df, [:block_number])

    #Find earliest tx time
    minTxTime, ind = findmin(df.createdt)

    #Create columns times starting from the earliest scenario tx and block
    df[:txcreated_norm] = 1
    df[:blockcreated_norm] =  1
    for r in eachrow(df)
        r[:txcreated_norm] = r[:createdt] - minTxTime
        r[:blockcreated_norm] = r[:block_createdt] - df.block_createdt[1]
    end

    #Get number of tx's
    rows = nrow(df)

    #Group tx's by time (second) and calculate peak, mean values
    txGroupedByTime = by(df, :txcreated_norm, N = :txcreated_norm => length)
    maxTime, ind = findmax(txGroupedByTime[:txcreated_norm])
    maxTime = maxTime+1;
    meanTPS = mean(txGroupedByTime[:N])
    peakTPS, row = findmax(txGroupedByTime[:N])

    #Group tx's by Block (second) and calculate peak, mean values
    txGroupedByBlockNum = by(df, :block_number, N=:block_number => length)
    blocks = nrow(txGroupedByBlockNum)
    meanTPB = mean(txGroupedByBlockNum[:N])
    peakTPB, row = findmax(txGroupedByBlockNum[:N])
    minTPB, row = findmin(txGroupedByBlockNum[:N])

    #Compute block stats
    blocksGroupedByBlockTime = unique(by(df, :block_number, x -> [x.blockcreated_norm][1]))
    blocksGroupedByBlockTime = by(blocksGroupedByBlockTime, :x1, N = :block_number => length )
    meanBPS = mean(blocksGroupedByBlockTime[:N])
    peakBPS, row = findmax(blocksGroupedByBlockTime[:N])

    #Generate response string
    html_text=html_text*"Average TPS:$meanTPS;"
    html_text=html_text*"Peak TPS:$peakTPS;"
    html_text=html_text*"Total Blocks:$blocks;"
    html_text=html_text*"Average BPS:$meanBPS;"
    html_text=html_text*"Peak BPS:$peakBPS;"
    html_text=html_text*"Mean TPB:$meanTPB;"
    html_text=html_text*"Peak TPB:$peakTPB;"
    html_text=html_text*"Min TPB:$minTPB;"
    html_text=html_text*"TXs:$rows;"
    html_text=html_text*"Max Time:$maxTime"
    html_text
end

#Add function to root route
route("/", AnalyzeBCData)

#Start server
Genie.AppServer.startup()
```

Bibliography

1. U.S. DOT - ITS/JPO, "USDOT's Intelligent Transportation Systems (ITS) Strategic Plan 2015-2019," p. 96, 2014. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/3506 [Accessed: 2018-08-10]

2. A. Sumalee and H. W. Ho, "Smarter and more connected: Future intelligent transportation system," *IATSS Research*, vol. 42, no. 2, pp. 67–71, 2018. [Online]. Available: https://doi.org/10.1016/j.iatssr.2018.05.005

3. U.S. Government Accountability Office (GAO), "Intelligent Transportation Systems' Promise for Managing Congestion Falls Short, and DOT Could Better Facilitate Their Strategic Use," Tech. Rep. September, 2005. [Online]. Available: https://www.gao.gov/assets/250/247752.pdf

4. S. M. Hussain, T. S. Ustun, P. Nsonga, and I. Ali, "IEEE 1609 WAVE and IEC 61850 Standard Communication Based Integrated EV Charging Management in Smart Grids," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7690–7697, 2018.

5. "IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages," *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pp. 1–240, 2016. [Online]. Available: https://doi.org/10.1109/5.771073

6. T. Derenge, "FCC ALLOCATES SPECTRUM IN 5.9 GHz RANGE FOR INTELLIGENT TRANSPORTATION SYSTEMS USES," p. 2, oct 1999. [Online]. Available: https://docs.fcc.gov/public/attachments/DOC-177370A1.pdf [Accessed: 2018-12-26]

7. "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services," *IEEE Std 1609.3-2010 (Revision of IEEE Std 1609.3-2007)*, pp. 1–144, 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5680697/

8. J. W. Connors, "Assessing the Competing Characteristics of Privacy and Safety within Vehicular Ad Hoc Networks," Ph.D. dissertation, Air Force Institute of Technology, 2018. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/1055987.pdf

9. B. Cronin, "Vehicle Based Data and Availability," U.S.D.O.T, ITS/JPO, Tech. Rep., 2012. [Online]. Available: https://www.its.dot.gov/itspac/october2012/PDF/data_availability.pdf

10. “5.9 GHz DSRC Connected Vehicles for Intelligent Transportion Systems,” Tech. Rep., 2013. [Online]. Available: https://ecfsapi.fcc.gov/file/7520943378.pdf

11. K. C. Bentjen, “Mitigating The Effects of Cyber Attacks and Human Control in an Autonomous Intersection,” Ph.D. dissertation, Air Force Institute of Technology, 2018. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/1055973.pdf

12. I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, “Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study,” in *Proceedings of the 19th USENIX Conference on Security.* USENIX Association, 2010, p. 21. [Online]. Available: https://www.usenix.org/legacy/event/sec10/tech/full_papers/Rouf.pdf

13. F. M. Benčić and I. P. Žarko, “Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph,” *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1569–1570, 2018. [Online]. Available: http://arxiv.org/abs/1804.10013

14. S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” Tech. Rep., 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

15. L. Lamport, R. Shostak, and M. Pease, “The Byzantine Generals Problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982. [Online]. Available: http://portal.acm.org/citation.cfm?doid=357172.357176

16. L. Baird, “The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance,” Swirlds, Tech. Rep., 2016. [Online]. Available: https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

17. L. Baird, M. Harmon, and P. Madsen, “Hedera: A Governing Council & Public Hashgraph Network,” pp. 1–27, 2018. [Online]. Available: https://s3.amazonaws.com/hedera-hashgraph/hh-whitepaper-v1.1-180518.pdf

18. A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak, “BlockChain: A Distributed Solution to Automotive Security and Privacy,” *IEEE Communications Magazine*, vol. 55, no. 12, pp. 119–125, 2017.

19. S. Popov, “The Tangle,” 2018. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf [Accessed: 2018-07-25]

20. M. Pilkington, “Blockchain Technology: Principles and Applications,” *Research Handbook on Digital Transformations, edited by F. Xavier Olleros and Majlinda Zhegu. Edward Elgar, 2016*, pp. 225–246, 2016. [Online]. Available: https://ssrn.com/abstract=2662660

21. V. Buterin, “On Public and Private Blockchains,” Web, 2015. [Online]. Available: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/ [Accessed: 2018-08-08]

22. J. O’Connell, “What Are the Use Cases for Private Blockchains? The Experts Weigh In,” 2016. [Online]. Available: https://bitcoinmagazine.com/articles/what-are-the-use-cases-for-private-blockchains-the-experts-weigh-in-1466440884/ [Accessed: 2019-01-08]

23. 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, no. 6, pp. 3154–3164, 2017.

24. Y. Yuan and F.-Y. Wang, “Towards blockchain-based intelligent transportation systems,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 2663–2668. [Online]. Available: http://ieeexplore.ieee.org/document/7795984/

25. Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, “Blockchain Challenges and Opportunities : A Survey Shaoan Xie Hong-Ning Dai Huaimin Wang,” *International Journal of Web and Grid Services*, pp. 1–24, 2016. [Online]. Available: http://inpluslab.sysu.edu.cn/files/blockchain/blockchain.pdf

26. M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” in *Proceeding OSDI ’99 Proceedings of the third symposium on Operating systems design and implementation*, no. February. New Orleans, LO, USA: USENIX Association Berkeley, 1999, pp. 173–186.

27. Hyperledger Architecture Working Group, “Hyperledger Architecture, Volume 1,” p. 15, 2017. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf [Accessed: 2018-06-22]

28. K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

29. Y. Kopylova, C. Farkas, and W. Xu, “Accurate accident reconstruction in VANET,” in *Li Y. (eds) Data and Applications Security and Privacy XXV. DBSec 2011. Lecture Notes in Computer Science*, vol. 6818 LNCS, 2011, pp. 271–279. [Online]. Available: http://dl.ifip.org/db/conf/dbsec/dbsec2011/KopylovaFX11.pdf

30. M. Singh and S. Kim, “Blockchain Based Intelligent Vehicle Data sharing Framework,” *CoRR*, 2017. [Online]. Available: http://arxiv.org/abs/1708.09721

31. C. Oham, S. S. Kanhere, R. Jurdak, and S. Jha, “A Blockchain Based Liability Attribution Framework for Autonomous Vehicles,” Tech. Rep., 2018. [Online]. Available: https://arxiv.org/pdf/1802.05050.pdf

32. M. Cebe, E. Erdin, K. Akkaya, H. Aksu, and S. Uluagac, “Block4Forensic: An Integrated Lightweight Blockchain Framework for Forensics Applications of Connected Vehicles,” *IEEE Communications Magazine*, vol. 56, no. 10, pp. 50–57, 2018. [Online]. Available: https://arxiv.org/pdf/1802.00561.pdf

33. C. Oham, R. Jurdak, S. S. Kanhere, A. Dorri, and S. Jha, “B-FICA: BlockChain based Framework for Auto-insurance Claim and Adjudication,” University of New South Wales, CSIRO Data61, Tech. Rep., 2018. [Online]. Available: https://arxiv.org/pdf/1806.06169.pdf

34. G. Greenspan, “MultiChain Private Blockchain-White Paper,” Tech. Rep., 2015. [Online]. Available: https://www.multichain.com/download/MultiChain-White-Paper.pdf [Accessed: 2019-01-04]

35. 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. Vukolić, S. W. Cocco, and J. Yellick, “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains,” in *EuroSys 18: Thirteenth EuroSys Conference 2018.* Porto, Portugal: ACM, 2018, p. 15. [Online]. Available: https://arxiv.org/abs/1801.10228http://dx.doi.org/10.1145/3190508.3190538

36. C. Cachin, “Architecture of the Hyperledger Blockchain Fabric,” Tech. Rep., 2016. [Online]. Available: https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf

37. P. Thakkar, S. Nathan, and B. Vishwanathan, “Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform,” 2018. [Online]. Available: http://arxiv.org/abs/1805.11390

38. The Linux Foundation, “Hyperledger Fabric (1.2) Glossary,” 2018. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/glossary.html [Accessed: 2018-12-06]

39. C. H. Papadimitriou and P. C. Kanellakis, “On Concurrency Control by Multiple Versions,” in *PODS ’82 Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems.* ACM, 1982, pp. 1–7. [Online]. Available: papers://2cd573f8-44b1-4938-8484-cf0e6dcd735b/Paper/p891

40. “Hyperledger Composer,” 2018. [Online]. Available: https://hyperledger.github.io/composer/latest/ [Accessed: 2018-12-06]

41. “Hyperledger Explorer,” 2018. [Online]. Available: https://github.com/hyperledger/blockchain-explorer

42. M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, *SUMO-Simulation of Urban MObility An Overview.* [Online]. Available: http://sumo.dlr.de/pdf/simul_2011_3_40_50150.pdf

43. The Linux Foundation, “Bringing up a Kafka-based Ordering Service,” 2018. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/kafka.html [Accessed: 2018-12-07]

44. NHTSA, “Odometer Fraud,” 2010. [Online]. Available: https://www.nhtsa.gov/equipment/odometer-fraud [Accessed: 2019-01-16]

45. D. Corum, “Insurance Research Council Finds That Fraud and Buildup Add Up to $7.7 Billion in Excess Payments for Auto Injury Claims,” Insurance Research Council, Tech. Rep., 2015. [Online]. Available: https://www.insurancefraud.org/downloads/InsuranceResearchCouncil02-15.pdf

46. “The State of Insurance Fraud Technology,” Coalition Against Insurance Fraud, Tech. Rep., 2016. [Online]. Available: https://www.insurancefraud.org/downloads/State_of_Insurance_Fraud_Technology2016.pdf

47. N. Nachar, “The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution,” Tech. Rep. 1, 2008. [Online]. Available: http://www.tqmp.org/RegularArticles/vol04-1/p013/p013.pdf

48. P. A. Bernstein and N. Goodman, “Multiversion concurrency control—theory and algorithms,” *ACM Transactions on Database Systems*, vol. 8, no. 4, pp. 465–483, 1983. [Online]. Available: http://portal.acm.org/citation.cfm?doid=319996.319998

49. A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, “MedRec: Using blockchain for medical data access and permission management,” *Proceedings - 2016 2nd International Conference on Open and Big Data, OBD 2016*, pp. 25–30, 2016.

50. K. Bentjen, S. Graham, and S. Nykl, “Modelling Misbehaviour in Automated Vehicle Intersections in a Synthetic Environment,” *ICCWS 2018 13th International Conference on Cyber Warfare and Security*, 2018.

51. J. Sousa, A. Bessani, and M. Vukolić, “A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform,” no. 1, pp. 51–58, 2017. [Online]. Available: http://arxiv.org/abs/1709.06921

52. Y. Qian, K. Lu, and N. Moayeri, “A secure VANET MAC protocol for DSRC applications,” in *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*. New Orleans, LO, USA: IEEE, 2008, pp. 1945–1949.

53. “HLF Private Data,” 2018. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.2/private-data/private-data.html [Accessed: 2018-10-29]

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 21–03–2019	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2017 — Mar 2019

4. TITLE AND SUBTITLE Modeling a Consortium-based Distributed Ledger Network with Applications for Intelligent Transportation Infrastructure	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Cintron, Luis A., Capt, USAF	**5d. PROJECT NUMBER** 18G230 **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering an Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-18-M-019
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory 2241 Avionics Circle WPAFB OH 45433-7765 Attn: Steven Stokes COMM 937-528-8035 Email: steven.stokes@us.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RYWA **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Emerging distributed-ledger networks are changing the landscape for environments of low trust among participating entities. Implementing such technologies in transportation infrastructure communications and operations would enable, in a secure fashion, decentralized collaboration among entities who do not fully trust each other. This work models a transportation records and events data collection system enabled by a Hyperledger Fabric blockchain network and simulated using a transportation environment modeling tool. A distributed vehicle records management use case is shown with the capability to detect and prevent unauthorized vehicle odometer tampering. Another use case studied is that of vehicular data collected during the event of an accident. It relies on broadcast data collected from the Vehicle Ad-hoc Network (VANET) and submitted as witness reports from nearby vehicles or road-side units who observed the event taking place or detected misbehaving activity. The experimental testbed shows that Hyperledger Fabric and other distributed ledger technologies hold promise for the collection of transportation data and the collaboration of applications and services that consume it.

15. SUBJECT TERMS

Intelligent Transportation Systems, Distributed Ledger Technologies, Blockchain, V2X Communication, VANET

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Scott Graham, AFIT/ENG
U	U	U	UU	160	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-6565 x4581; scott.graham@afit.edu