Secured Scalable Blockchain Networks for Trustworthy Distributed Deep Learning in VANETs

by

Zhaowei Ma

A dissertation submitted to Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Information Technology

Carleton University
Ottawa, Ontario

© 2023
Zhaowei Ma
Abstract

Distributed deep learning (DDL) within vehicular ad hoc networks (VANETs) holds profound significance in developing smart applications, such as intelligent transport systems and autonomous driving, where multiple parties are coordinated to leverage the training capability and acquisitive intelligence. Blockchain (BC) is a distribution technology promising for trustworthy DDL, holding the divide-and-conquer concept with decentralized management and consensus algorithms to reduce the exposure of sensitive controllers and thus the risks of malicious system-wide attacks. Moreover, zero trust architecture (ZTA) concepts are promoted as innovative cybersecurity solution which can be integrated with BC to address the resource-limited and infrastructure-less issues to augment the strength of DDL in VANETs. In this dissertation, the BC and ZTA potential is explored to construct reliable VANETs for trustworthy data sharing and thus to support significant within-VANET DDL and relevant applications. Firstly, virtualized distributed ledger technology (vDLT) is developed as the multimedia BC platform, followed by vDLT-based VANETs built to solve the unsteady communication; secondly, vDLT is improved to transmit and secure traffic events in VANETs; subsequently, a vDLT-based DDL system is proposed to enhance object detection (OD) inside VANETs; finally, ZTA and sharding scheme are enabled in vDLT-based VANETs for improved protection. Generally, vDLT runs with virtualized resource and sharding supports for scalability improvement, along with multi-layered consensus algorithm and an adaptable hierarchical and decentralized PBAC (hdPBAC) access control
model to agilely protect the DDL procedures and relevant DDL-related applications. The proposed system has been developed, including the vDLT system, the multimedia streaming over vDLT in fixed networks and VANETs, a precursory vDLT-based DDL system for OD purpose, and the integration of ZTA into scalable-BC-based VANETs. The evaluation results show the feasibility of proposed design and demonstrate the significance of performance improvements.
Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor, Prof. Richard Yu, who have been working with his wholehearted dedication to supervise my PhD study. His immense knowledge and cutting-edge insights have been encouraging me of elevating my academic research quality. He have also endeavored to find fundings to support my study and daily life to keep my work proceeding smoothly. Without his invaluable advice and continous supports, the achievements during my study would have been impossible. Words are powerless to express my gratitude to him, and I will maintain my grateful heart constantly and sincerely.

I am also grateful to Prof. Omair Shafiq for his valuable comments and suggestions on my research and his knowledge and experience in engineering shared for my improvements. Additionally, I would like to thank Prof. Ashraf Matrawy, Prof. Rong Liu, and Prof. Azzedine Boukerche for serving in my dissertation defence committee. Special thanks are given to my friends and my previous lab mates, Dr. Dajun Zhang, Dr. Jie Feng and Dr. Li Zhu, for the great cooperation in project implementation and their constructive suggestions on my research work.

Finally, I would like to express my gratitude to my family. Their tremendous understanding and encouragement stimulates me into confidently pursuing my objectives. I am thankful to have a big family who always gives me warm caring and keeps me positive and powerful in my life.
# Contents

Abstract i

Acknowledgements iii

List of Figures vii

List of Tables x

List of Abbreviations xii

1 Introduction 1

1.1 Background ........................................ 1
1.2 Problem Statement .................................... 2
1.3 Proposed Solutions .................................... 5
1.4 Dissertation Structure ................................. 8

2 Literature Review 9

2.1 Distribution over VANETs .............................. 9
2.2 DDL with Centralized Management ...................... 12
2.3 BC-enabled DDL ........................................ 16
2.4 Security Solution in BC-based VANETs .............. 20
2.5 Research Gaps .......................................... 23
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Proposed Design</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Overview</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>Research Questions</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Contributions and Publications</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Multimedia vDLT and vDLT-based VANETs</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Multimedia Networking via vDLT</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Distributed VANETs based on vDLT</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>Sharing Visual Traffic Information via vDLT</td>
<td>74</td>
</tr>
<tr>
<td>5.1</td>
<td>System Design</td>
<td>75</td>
</tr>
<tr>
<td>5.2</td>
<td>System Development</td>
<td>82</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiment and Evaluation</td>
<td>91</td>
</tr>
<tr>
<td>5.4</td>
<td>Remarks</td>
<td>99</td>
</tr>
<tr>
<td>6</td>
<td>DDL Based on vDLT with 6G Features</td>
<td>102</td>
</tr>
<tr>
<td>6.1</td>
<td>System Design</td>
<td>102</td>
</tr>
<tr>
<td>6.2</td>
<td>Problem Formulation</td>
<td>110</td>
</tr>
<tr>
<td>6.3</td>
<td>Simulation and Evaluation</td>
<td>118</td>
</tr>
<tr>
<td>6.4</td>
<td>Remarks</td>
<td>127</td>
</tr>
<tr>
<td>7</td>
<td>ZTA for Sharding-scaled BC-based VANETs</td>
<td>129</td>
</tr>
<tr>
<td>7.1</td>
<td>Framework Architecture</td>
<td>129</td>
</tr>
<tr>
<td>7.2</td>
<td>System Development</td>
<td>138</td>
</tr>
<tr>
<td>7.3</td>
<td>System Evaluation</td>
<td>149</td>
</tr>
<tr>
<td>7.4</td>
<td>Remarks</td>
<td>157</td>
</tr>
<tr>
<td>8</td>
<td>Conclusion and Future Works</td>
<td>158</td>
</tr>
<tr>
<td>8.1</td>
<td>Conclusion</td>
<td>158</td>
</tr>
<tr>
<td>8.2</td>
<td>Future Works</td>
<td>159</td>
</tr>
</tbody>
</table>
List of Figures

3.1 The proposed vDLT-based DDL in VANETs. ..................... 26

4.1 The architecture of proposed system that protects surveillance recordings via BC technology in alliance with cloud computing. ......... 36
4.2 The design of video receiving and processing procedures. ............. 44
4.3 The design of MDS service and involved SCs in vDLT. ................. 46
4.4 Comparison between frames associated with different fingerprints. ... 52
4.5 The average throughput at different time points. .................... 53
4.6 Comparison of time sequence in different producers. ................. 54
4.7 Comparison of time latency in different producers. .................. 54
4.8 Comparison between frames associated with different fingerprints. ... 55
4.9 The architecture of VANETs based on vDLT. ....................... 58
4.10 The components and their relationship of vDLT-based VANETs. ... 61
4.11 A general structure of BC used in VANETs. ....................... 63
4.12 Delay comparison between AODV, DSDV and the proposed scheme. ... 69
4.13 Throughput comparison between AODV, DSDV and the proposed scheme. 70
4.14 Average packet loss ratio comparison between AODV, DSDV and the proposed scheme. ....................... 71

5.1 The proposed system architecture composed of a VANET and a BC Network ....................................................... 76
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>The 3-tires model of proposed system</td>
<td>79</td>
</tr>
<tr>
<td>5.3</td>
<td>The services for collaboratively discovering BC access points</td>
<td>83</td>
</tr>
<tr>
<td>5.4</td>
<td>The services for uploading traffic information via BC access points</td>
<td>85</td>
</tr>
<tr>
<td>5.5</td>
<td>The services for downloading traffic information via BC access points</td>
<td>88</td>
</tr>
<tr>
<td>5.6</td>
<td>The set of collaborative SCs for traffic event sharing</td>
<td>89</td>
</tr>
<tr>
<td>5.7</td>
<td>A scenario of uploading and downloading through BC platform</td>
<td>95</td>
</tr>
<tr>
<td>5.8</td>
<td>The throughput of receiving videos in BC access points</td>
<td>96</td>
</tr>
<tr>
<td>5.9</td>
<td>The time latency evaluated in BC access points and vDLT</td>
<td>97</td>
</tr>
<tr>
<td>5.10</td>
<td>The efficiency of producing frame blocks in vDLT</td>
<td>98</td>
</tr>
<tr>
<td>6.1</td>
<td>The overall architecture of proposed system in which the highlighted</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>MBS is currently creating blocks of aggregated models</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>The process of model aggregation via BC system</td>
<td>106</td>
</tr>
<tr>
<td>6.3</td>
<td>The designs of relational SCs assisting DDL</td>
<td>109</td>
</tr>
<tr>
<td>6.4</td>
<td>The number of blocks for an aggregated model</td>
<td>120</td>
</tr>
<tr>
<td>6.5</td>
<td>The volume level of transactions for parameter sharing</td>
<td>121</td>
</tr>
<tr>
<td>6.6</td>
<td>The latency comparison between the proposed system and two baselines</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>in terms of different number of DDL workers</td>
<td></td>
</tr>
<tr>
<td>6.7</td>
<td>The accuracy of proposed system in terms of SBS-DDL provision</td>
<td>123</td>
</tr>
<tr>
<td>6.8</td>
<td>The loss of proposed system in terms of SBS-DDL provision</td>
<td>123</td>
</tr>
<tr>
<td>6.9</td>
<td>Comparison of loss performance in training stages</td>
<td>124</td>
</tr>
<tr>
<td>6.10</td>
<td>Comparison of loss performance in validation stages</td>
<td>125</td>
</tr>
<tr>
<td>6.11</td>
<td>Comparison of accuracy in training stages</td>
<td>125</td>
</tr>
<tr>
<td>6.12</td>
<td>Comparison of accuracy in validation stages</td>
<td>126</td>
</tr>
<tr>
<td>7.1</td>
<td>The general architecture of proposed system which enables ZTA and</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>BC sharding to jointly improve security, decentralization and scalability of BC-based VANETs</td>
<td></td>
</tr>
</tbody>
</table>
7.2 The major process and components of system for managing data sharing in VANETs through ZTA and BC sharding, which can be continually iterated to achieve adaptation. ........................................ 134

7.3 The proposed SCs and their relations for access control. ................. 135

7.4 Device security status represented in images where the left device is more attractive to attackers .................................................. 140

7.5 Hierarchical and decentralized PBAC model in BC system. ............ 143

7.6 The losses of training deep autoencoder. ....................................... 151

7.7 The distances in K-Means clusters. ............................................. 152

7.8 The latency of security indexing via transactions. ........................ 153

7.9 The throughput produced in security indexing via transactions. ...... 154

7.10 The average latency in transaction execution. ............................. 155

7.11 The success rates of transaction sharing. ..................................... 156
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Tools used to construct the project.</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Specifications for system evaluation.</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Configuration of SCs for procedural data exchanging.</td>
<td>64</td>
</tr>
<tr>
<td>4.4</td>
<td>Configuration of the platform for experiment.</td>
<td>66</td>
</tr>
<tr>
<td>5.1</td>
<td>Configuration of components for simulation.</td>
<td>92</td>
</tr>
<tr>
<td>6.1</td>
<td>The list of major components configured.</td>
<td>119</td>
</tr>
<tr>
<td>6.2</td>
<td>An example of simulated connection between SBSs and edge devices.</td>
<td>119</td>
</tr>
<tr>
<td>7.1</td>
<td>The list of devices and related configuration.</td>
<td>150</td>
</tr>
</tbody>
</table>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAC</td>
<td>attribute-based access control</td>
</tr>
<tr>
<td>BC</td>
<td>blockchain</td>
</tr>
<tr>
<td>BS</td>
<td>base station</td>
</tr>
<tr>
<td>CC</td>
<td>cloud computing</td>
</tr>
<tr>
<td>DDL</td>
<td>distributed deep learning</td>
</tr>
<tr>
<td>DDQN</td>
<td>double deep Q network</td>
</tr>
<tr>
<td>hdPBAC</td>
<td>hierarchical and decentralized PBAC</td>
</tr>
<tr>
<td>IoE</td>
<td>Internet of everything</td>
</tr>
<tr>
<td>IPFS</td>
<td>InterPlanetary file system</td>
</tr>
<tr>
<td>MBS</td>
<td>macro base station</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov decision process</td>
</tr>
<tr>
<td>MDS</td>
<td>multimedia data sink</td>
</tr>
<tr>
<td>ML</td>
<td>machine learning</td>
</tr>
<tr>
<td>NFV</td>
<td>network function virtualization</td>
</tr>
</tbody>
</table>
OD .......................... object detection

PBAC ........................ policy-based access control

PBFT ........................ practical Byzantine fault tolerance

RBAC ........................ role-based access control

RSU .......................... road side unit

SBS .......................... small base station

SC .............................. smart contract

SDN ............................ software-defined networking

SGD ............................ stochastic gradient descent

uRLLC ........................ ultra-reliable and low-latency communication

VANET ........................ vehicular ad hoc network

vDLT ............................ virtualized distributed ledger technology

ZTA ............................. zero trust architecture
Chapter 1: Introduction

1.1 Background

DDL systems have been widely implemented by cooperating multiple networking machines. DDL in local machines using the internal GPUs or TPUs and PCIe lanes [122] becomes insufficient for ever-growing networks, such that it is more efficient to apply the networking DDL which employs a group of remote machines for better scalability and capability [20] [10]. Advanced network technologies, such as clustering and cloud computing (CC), have shown outstanding strength in improving DDL [120], but their common drawbacks are witnessed in the highly dependence on centralized controllers.

The advance of DDL in heterogeneous networks is being beneficial from BC technology. The divide-and-conquer principle of BC guarantees the trustworthy data storage and communication as data with its sequence in BC is strictly protected via the multi-parties confirmation [78] [129]. Autonomous consensus and randomized government is able to prompt the integrity and privacy during the DDL process [91]; the removal of complex centralized management can promise high scalability and efficiency of parameter sharing in DDL [135] [21]; the flexible architecture of BC helps relieve the collaboration overhead between DDL workers. BC technology opens up an innovative direction of improving DDL design and hence the performance of DDL-related applications.

VANETs armed with DDL supports can enhance the transportation management and road safety protection. Various smart applications are being designed into VANETs, such as surveillance inside Smart Cities, Virtual Reality, gamification and Autonomous Driving, and commonly the intelligence acquisition is tightly related to OD [134] [93] [4]. As the fundamental support, the performance of OD should be improved to empower the intelligence in VANET, and accordingly DDL can coordinate
multiple VANET parties, even the weak devices, to leverage the training capability and acquisitive knowledge for an ensemble and enhanced trained model [66] [35]; therefore, DDL is believed to benefit the enhancement of OD in VANETs and thus the higher-level applications.

The VANETs to apply DDL urge remarkable improvements of both reliability and efficiency, which can be achieved by the synthesis of BC and ZTA technologies. For a better trained model, the volume of data is exploding along with the scale of training networks promptly expanding [2] [3]. In VANETs, networking DDL solutions using centralized components aggregate various issues, such as the constrained resource efficiency and communication reliability [136] [38] [14]. BC is believed to empower the DDL enhancement despite challenges [55] [55], but BC-based VANETs are faced with various cybersecurity issues, such as dishonest transactions causing system disruption [33] [54] [34] and on-site attacking vehicle controlling multiple victims [143]. ZTA which supports adaptation in protecting both security and trust are found effective to establish comprehensive protection into BC-based VANETs [109] [98]. ZTA eliminates network boarder constraints and continuously monitor network-wide status to adjust policies for enhanced protection, and meanwhile runs a policy engine for agile algorithms [107]. Therefore, ZTA-based security solutions is promising for the improvement of BC-based DDL in VANETs.

1.2 Problem Statement

The performances of reliability and efficiency are the major concerns in the DDL improvement. However, trade-off between reliability and efficiency exists, where reliability requires time spent on complex management, while adding resource to improve efficiency will lead to extra management, and it is inevitable in designing DDL schemes. Especially in VANETs, the conflicts between resource provisions and DDL resource requirements are outstanding. Furthermore, as a typical DDL application
in VANETs, OD which is used to protect traffic safety is a dynamic and real-time process which continuously collects the on-the-scene information, and it means that the inputs of DDL as well as their source require to be protected for solid trust and integrity of data. Overall, only with trustworthy inputs, parameters, models and operations among the DDL participants, the DDL and thus the OD process is secured and reliable.

Actually, there are two types of DDL designs prevail. Local-machine DDL sacrifies scalability and reliability for efficiency and it becomes deficient and helpless in handling the continuously increasing workload. A standalone machine is protected by limited strategies, whereas the proliferation of protection is resource-consuming. It is unrealistic for a machine to pre-install infinite computing resource or to dynamically increase hardware during a running DDL process. CC-based DDL runs centralized orchestration which implies vulnerability. It is risky to centralize the supreme privilege in individual entities. Modification of the CC controllers for DDL could influence the exiting workflow. Vandalizing the controllers or tampering the global parameter storage is effective to modify the training procedures. Even if controllers can be clustered, attacking individuals is possible to destroy their entire integrity. Thus, malicious attacks on the key parts of CC platforms can result in a counterfeited model. Besides, the expansion of DDL group causes considerable latency in the CC cores. Overall, CC-based DDL is heavily dependent on the design of centralized management.

BC technology addresses the concerns about centralization, but its distributed consensus and validation procedures require frequent and timely synchronization of data in a large group of participants. Without global storage in BC networks, data is broadcast among participants over secured P2P links; thus, the topology of BC networks impacts the reliability and efficiency of data sharing. On receiving broadcasting data, a group of participants run validation and consensus on each data copy; the more participants the more secured can be the data, but the latency in these stages is accordingly increased. Besides, the latency is intensely impacted by
the size of data block, and consequently without optimization, excessive latency will lead to failed block synchronization and thus ruin the data integrity. In VANETs, the resource limitation and infrastructure-less topology challenges the deployment of BC system, such that integration of BC into VANETs requires in-depth designs of system mechanism and complex interfaces at the goal of improving both efficiency and reliability.

As networking applications, both BC and VANET systems are inevitably faced with cybersecurity concerns which consequently congregate in the BC-based VANETs applications. Traditional perimeter-based solutions become unsatisfactory confronting the intricate attack vectors in BC-based VANETs. For VANETs, it is impractical to install full-size firewalls and designate various zones due to the resource limitation and device mobility, and for BC networks, trustless nodes and stochastic alternation of participant roles make it incompetent to plan constant sites and policies. In progress, ZTA is being promoted to overcome the drawbacks of traditional security approaches to meet the ever-growing networks, and hence is promising for BC-based VANETs. However, the fusion of ZTA and BC sacrifices system performance and unavoidably imposes overload in VANETs, such that it is demanding to improve the scalability of BC-based VANETs for efficient protection. But still, the involvement of heterogeneous data and devices in VANETs challenges the expectation of scaling solutions, and complicated scaling algorithms will even reduce the system performance.

In general, DDL and OD designs in VANETs can be reinforced via BC and ZTA technologies, but it demands to significantly improve both reliability and efficiency in process. For the improvements, a series of factors, such as security access control, scaling method, consensus mechanism, participant robustness and trust management are necessitated. More precisely, to construct the proposed idea, there are a series of problems need to be carefully addressed.

1. What are the advantages of distribution in VANETs which are formed based on
unstable connections?

2. Why can the BC-based DDL which is autonomously managed be more reliable than the DDL designs which is controlled by the centralized authority?

3. What issues are in current BC-based DDL schemes, especially for OD purposes, and what can be improved?

4. Who can share, train or protect the information in BC networks in OD procedures, and Who is responsible for managing the process?

5. What information is needed to prove and protect the trust of OD since management is alternated over various BC participants?

6. Whose privacy should be protected, and what information should be concealed during the DDL process to prevent data from being vandalized by trustless participants?

7. What are the security concerns with BC-based VANETs? Why the traditional security solutions are impractical?

8. What are the practical options for improving the scalability of BC system in VANETs and thus the performance of BC-based VANETs?

1.3 Proposed Solutions

A secured and scaled vDLT-based DDL system in VANETs is proposed to address the aforementioned issues. The proposed system is focused on protecting the heterogeneous data sharing in support of DDL and improving the security and efficiency in DDL process. The proposed system is able to spread training tasks over numerous BC participants, in which vDLT is developed as the BC system. It allows sharing the data imperative for DDL applications, such as traffic events, device attributes, model
parameters and security control policies, supports cooperative OD which is developed as a DDL application performed among active BC learners and is managed and protected by proposed consensus algorithms, executes sharding solutions and security access control models to swiftly protect the data and DDL procedures, and penetrates the functions of fusing and validating data including dataset features, DDL parameters, security attributes and related data components where are executed by the current BC producers and subsequently validated by the trusted witness entities in vDLT.

More precisely, the proposed system incorporates a series of advanced features to address the issues above. The major elements are listed as follows.

- Virtualization technology in vDLT improves the system scalability and security. Resource is virtualized to allow the dynamical increase of computing capability; virtualized environment prompts the recovery from application failure and provides sandbox protection; overlay network is created to secure and stabilize the distributed architecture for frequent data sharing in VANETs.

- Smart contracts (SCs) are the protocol for managing training tasks and sharing parameters in vDLT networks. SCs are user-oriented and designed based on a complex privilege hierarchy, such that privacy of users and training contexts can be protected. Besides, the improvement of SC granularity can further elevate the system scalability.

- Asymmetrical cryptography and cryptographical hash functions are utilized for authentication protection, which respectively prevent the disclose of user private credentials and the tamper with user data. Together, they ensure the immutability of parameter sharing and state coordination during the DDL processes.

- Layered-Chain design is proposed to mirror the neural network structure into the vDLT system. Each layer forms a branch of chain, and cross-chain communication
allows parameters to be shared for assembling models in upper layers and the top-down government of DDL tasks.

- Consensus algorithms provide double-folds trust protection on DDL context and orchestration for parameter fusion and the model integrity, in which the first is to verify the assignment of DDL tasks, and the other to guarantee the authentication of DDL models and OD information.

- Multimedia streaming via the vDLT networks is supported, such that the vDLT-based DDL is effective for real-time and precise OD in VANETs. The timely data synchronization in vDLT allows trained models to be spread over the whole network and the OD execution ubiquitous within the VANETs.

- ZTA access controls and BC sharding solutions are built in VANETs to address the BC trilemma issues and improve the trust of data-intensive applications in VANETs. hdPBAC is designed for dynamic access control over BC-based VANETs along with sharding scheme enabled according traffic situation and device security.

In general, the proposed system aims to improve DDL performance in the unsteady VANET architectures by taking the advantages of DDL and BC together with security solutions to comprehensively solve the issues of scalability, efficiency and reliability. The features above are proposed based on the DDL mechanism to increase the training capability and the vDLT framework to agilely scale up the training process, so as to promote and reinforce the reliable cooperation in model training and data sharing during the DDL process.
1.4 Dissertation Structure

The remainder of this proposal includes 8 chapters. Chapter 2 will investigate the related findings about distributed VANETs and wherein DDL designs which majorly pertain to the distribution feasibility in VANETs, the current development of DDL designs and the gap analysis. Chapter 3 will describe the research approach, focusing on the overall system architecture, research questions and the related actions. Chapter 4 will explain the first achievement, which is the construction of overlay network for vDLT communication. Chapter 5 will discuss the second contribution, which is the multimedia streaming supported in vDLT BC. Chapter 6 will analyze the preliminary design of vDLT-based DDL which is to solve typewriting recognition. Chapter 7 will present the design and development of sBC-V-ZTA framework which embraces adjustable access control model and sharding scheme for security improved in BC-based VANETs. Chapter 8 will conclude the research work on this proposed design and plan the future work for improvement on this proposal.
Chapter 2: Literature Review

2.1 Distribution over VANETs

Recently, the shift from optimization of routing protocols to implementation of distributed technology creates more opportunities to boost VANETs [111] [46] [5]. This evolution improves the collaboration between elements by logical arrangement instead of complex routing procedures. Successful examples are mainly witnessed in the integration of software-defined networking (SDN), CC, edge computing or even BC with VANETs, which achieve more energy saving, higher throughput, and solid data transmission.

The studies in [51] and [146] employ a centralized SDN controller, and distribute switches over multiple vehicles. As the inter-connection between vehicles is controlled through flow rules computed by a separate component, vehicles escape from complicated routing procedures. However, the locomotion of vehicles demands intensive but centralized computation of flow rules and excessive resource for updating flow tables. Moreover, the physical links have to change according to the update of flow tables.

Approaches in [13], [117] and [44] explore the architecture of CC in VANETs, where vehicles are grouped into a cloud, and there is a vehicle elected to as a broker or leader taking charge of the cloud structure. Hence, vehicles inside can reduce their routing overhead, and any vehicle in the cloud can be taken as provider. With the cloud, data delivery can be accelerated along with the reduction of routing request.

The solutions above count on a centralized component in the CC environment to establish the stable distribution. But since the VANET structure changes frequently, centralized controllers usually maintain the distribution at the cost of sacrificing reliability and scalability. Moreover, distribution enabled by BC which is built with decentralized ledger and consensus between participants, has proved solidity of data storage and efficiency of transmission.
Sharma et al. [106] introduce a BC-based vehicular network, where a centralized authority assigns some vehicles mining tasks and other infrastructures for relaying tasks. There are still some unsolved issues, such as unsteady connection amongst miner vehicles due to their locomotion, and unfair consensus between miners of centralized nomination. Qiu et al. [90] synthesize technologies of SDN, deep learning and BC in VANETs to predict and to protect the flow rules, and hence the integrity of data which utilized BC to validate the vehicle information and to assure that the data disseminated is trusted. But since the deep learning is centralized, the trust of flow rules is weak, and the spatial change of vehicles requires duplicated data to be propagated in multiple traffics.

Virtualization is also capable of solving the vulnerability issue in data transmission relying on the variable physical connection in VANETs, such that is able to consolidate the distribution over VANETs. The proposal in [57] spawn virtual access points in the physical eNodeB to simultaneously accept more vehicles. Upon spatial change of vehicles, a number of virtual eNodeBs will be created. With the help of a SDN controller, those moving vehicles can rapidly reach the dedicated eNodeB to obtain network resource. But still, SDN rules set for physical links and routing protocols between vehicles are required. On the contrary, the approach in [56] constructs the SDN-based virtual VANET where vehicles and road side units (RSUs) communicate via virtual IP address. The SDN controller counts on physical IP address, and the mapping between physical and virtual IP addresses is required for data transmission to follow SDN flow rules.

Furthermore, to support distributed training in VANETs, the design of content collection and sharing over distributed VANETs has gained much research interest in reliability improvement. Typically, traffic safety improvement is trending to use large-scale analysis of traffic data intelligence achievement. Chowdhury et al. [17] highlight the importance of traffic analysis for intelligent transportation system, and argue that data security issue influences the system capability. Ma et al. [69] develop
a neural network which trains a traffic speed prediction model based on official traffic information in order to improve driving safety. Lv et al. [68] propose their deep learning approach which applies trustworthy traffic data into training for accurate traffic flow prediction and hence for escorting road safety. With intelligence, VANETs are more secured and efficient to protect the transportation.

Shrestha et al. [108] present a system based on Bitcoin BC to secure the message exchange for VANETs. As considering the relationship between geographic location and traffic events, the authors design separate BCs for different countries to improve system efficiency. With BC consensus mechanism, the system aims at protecting the trust of traffic event messages exchanged between vehicles.

Ma et al. [75] design a VANET framework based on vDLT, which can enforce the reliable communication in VANETs with the immutability of BC and the scalability of virtual networks. In the framework, elements are connected via overlay networking, therefore, despite the physical resource is exhausted, data transmission can be maintained and consolidated by using BC to dynamically adjust virtual network resource and management.

Yang et al. [138] propose a BC-based system to validate and verify traffic event and its trust for improvement of VANETs. The system collects and analyzes the report of traffic events in VANETs, and then in real-time notifies vehicles the authentic traffic condition so as to improve traffic safety. The focus of this system is using consensus algorithm of BC to protect the trust and transmission of traffic events.

Ayaz et al. [6] employ BC to protect message security for VANETs, with the consideration of economic incentive which can encourage vehicles to report and protect the authentic traffic information. The authors propose an economic model which combines price and reputation of message dissemination to attract more efforts, and a consensus algorithm based on voting scheme to select and reward the trustworthy vehicles to defend attacks.

Ren et al. [94] design the intelligent transportation system in use of BC and
Internet-of-Things technologies in VANETs. The intelligence of traffic system is boosted through extensive collection and analysis of traffic information, therefore reliable data sharing in VANETs is the key support. The system uses BC to validate the payment of receiving trustworthy traffic information, and CC as back-end components to process and store traffic data.

In general, these schemes take vehicles as BC nodes to enlarge the area of data provisioning, but require numerous reconfiguration of the physical network, during the movement of vehicles. BC nodes are still tightly dependent on underlying resource, physical addresses and traffic circumstances, which considerably constrain the system performance.

2.2 DDL with Centralized Management

The distribution over VANETs makes it efficient to design the DDL system which employs the VANET nodes. Nowadays, DDL has been effectively implemented, and mostly focus on sharing knowledge of deep learning with the centralized management components and global storage. Tremendous researches indicate that applying centralized management in collaboration of sharing knowledge and arranging tasks can greatly alleviate the burden of deep learning.

Yun et al. [142] introduce an algorithm for parameter matrix completion in a parallel distributed manner to improve the performance of deep learning. This approach utilizes shared memory for global access to parameters by all the working nodes. Asynchronous stochastic gradient descent (SGD) supported by Multi-threaded MPI and Thread Building Blocks is used to update parameters, and hence network blocking can be avoided.

Heigold et al. [40] implement an multilingual approach based on distributed DNNs, where the function of extracting features is shared and language-specific classifiers are used. DistBelief is proposed to distribute computation across multiple machines to
enable parallelism and manage the communication, synchronization and data transfer. Downpour SGD, as an asynchronous approach, is designed to prevent network blocking and influence caused by other failed nodes.

Ooi et al. [87] introduce a distributed deep learning system, SINGA to improve the performance of training big models. This system supports synchronous and asynchronous training approaches, which respectively accelerate each training iteration and the convergence rate of learning. In this system, parameters are stored in a centralized server and accessed by all the training nodes, while these nodes are grouped up to contribute their parameters after each iteration.

Lim et al. [60] [61] propose an shared memory-based framework to allow parameters of deep learning to be accessed by multiple inter-connected machines, so as to accomplish the collaborative learning and to speed up the entire training process. As suggested, a separate server holds the shared memory, which can be remotely accessed by all the participants using dedicated drivers and interfaces via networks. Eventually, the parameters are exchanged between all these networking nodes through the shared memory.

Li et al. [58] apply a new all-reduce into neural networks by improving the parallelization in Spark. This advance uses a user interface to support the SGD vector exchange between participating nodes instead of purely broadcasting serialized objects in Spark; besides, it stores the updated vector in memory other than disk. Based on map-reduce in Spark, a master node collects the reduced vectors from all the slave nodes, applies all-reduce to the whole incomes, and return back the final result to the next training stage.

Swearingen et al. [115] introduce the Auto-Tuned Models (ATM) which works as a distributed, scalable and collaborative system and is able to improve automated machine learning. A shared rational database is used to store the dataset, hyper-parameters, trained model and the logic control for a single run, and a reward-based Selector and meta-modeling Tuner are implemented, which jointly allow the system
to quickly search the most reasonable model to conduct the further training without much human intervention.

Hardy et al. [37] demonstrate that their compression scheme, AdaComp, can preserve the accuracy of deep learning which is accomplished by edge devices, and meanwhile reduce the amount of data transferred in networks. Similarly, this proposal establishes a centralized server and distributed working nodes in the network. The server is in charge of managing the global parameters shared across edge devices, while the devices work on training partial models based on a subset of data, compress and upload the results to the server.

Yaseen et al. [140] create a CC system based on Spark to accomplish the DDL with an emphasis to maintain the high accuracy and precision, and meanwhile to improve the scalability and robustness of learning. A master node is responsible for managing the training process, collecting and diffusing partial trained models, and generating the final classifiers, along with the HDFS for data sharing, Kryo for object serialization and Map-Reduce for training collaboration.

Natu et al. [84] present an end-to-end distributed deep learning, EasyDist, to reduce the training time as well as to preserve the accuracy. Especially, this proposal aims to facilitate the design of distributed deep learning through the use of Keras and TensorFlow, which respectively help with the definition of trained model and support the distribution of model. The cluster of working nodes is provided by CC with a central server to store global shared parameters.

Pumma et al. [89] develop LMDBIO which is a plugin for Caffe to improve I/O performance of parallel distributed Deep Learning. Caffe uses lightning memory-mapped database (LMDB) to share information in a clustering environment. LMDBIO and indexing using B+ tree help alleviate the I/O burden when a burst of sequential accesses to the database. Shared memory is mapped with a buffer in the local memory, and the parallel training processes are scheduled to access the shared memory by multiple communicators via local buffers.
Ahn et al. [1] develop ShmCaff based on Caffe which is a virtual shared memory framework to support DDL. Remote direct memory access (RDMA) eliminates the communication between application and kernel, for data exchange, a parameter server is used for global sharing, and asynchronous SGD approach is used for updating parameters across multi-workers which run synchronous SGD. Shared memory buffer (SMB) is created in a centralized server for global parameter updates.

Choi et al. [16] improve the transfer learning scaling the trained model through analyzing the semantic relationship between previous model and feedback of its utilization in a global component. Each training iteration of creates the semantic relationship in triplets to indicate the divergence gap between classes in target dataset and in anchor set, and the global component solves the confusion of target classes according to the relationship, and based on resulted feedback adjusts the training.

Wang et al. [127] propose a platform, Nexus, to enable distributed learning in some popular frameworks and to improve the time efficiency of training. A coordinator is used to schedule the distribution and aggregation of model parameters. Shared memory supports Nexus clients and servers to exchange information, Fault tolerance and straggler mitigation for reliable communication, and RDMA to accelerate parameter transfer.

Wu et al. [133] present a method to analyze the trained model in use of a small dataset, whose points are organized via an explicable boundary tree (EBTree) in a global storage. To interpret the classification done in the model, that is to traverse the EBTree but not to access the internal structure of the model. The points of small dataset form the interpretable decision boundary, which separates two different classes. A translation module in the centralized position allows trained models to be reused and shared among various participants.

Zhuang et al. [148] improve transfer learning to make progress in DDL development by introducing a deep Autoencoder which is comprised of an embedding layer and a label encoding layer. Autoencoder is centralized to learn the weight and bias
of encoding and decoding processes. The trained parameters are derived in the consecutive learning, and the algorithm in embedding layer improves the similarity of data distributions in source- and target- domain.

Boehm et al. [15] introduce an approach of fitting SystemML into the Spark which uses Spark APIs to support seamless integration, and automatic compiling of machine learning (ML) algorithms. With the global view in Spark, checkpoint injection is applied to prevent repeated retrievals of data and to eliminate unnecessary rewrites, and to bring some salient features, such as supporting distributed matrix representation and dynamic recompilation.

Dong et al. [25] propose a deep learning library, TensorLayer, to facilitate the development of DDL in clustering environment. Especially, TensorLayer provides a workflow module to support their asynchronous scheduling and failure recovery for concurrent training tasks. GridFS is used as a blob backend and MongoDB as an sample indexer, to support model cross-validation and hyper-parameter optimization with information exchanged in a shared infrastructure.

Di et al. [23] design a framework via feature isomorphism discovery (TLFid) in a global storage to improve distributed training. This approach firstly extracts features from source and target domains, then searches the common feature structure of source- and target- domain datasets, and finally reconstructs feature representation to build a target classifier. The global embedding approach helps construct a feature correlation matrix, and a feature mapping function to rapidly share knowledge.

### 2.3 BC-enabled DDL

BC as another major distributed platform is capable of supporting distributed VANETs and DDL architectures, such that it is worth investigating the findings of DDL supported by BC, and correspondingly exploring the potentials of BC-enabled DDL in improving OD in VANETs. Since the intensive and large-scale content sharing in
VANETs using BC has been demonstrated effective in aforementioned analysis, these features are believed to benefit the DDL which is data-intensive application.

Rathore et al. [92] propose DeepBlockIoTNet, which utilizes BC networks to secure the DDL process for the IoT intelligence improvement, their design decentralizes DDL tasks into edge IoT devices along with using BC nodes to aggregate parameters, and protects the privacy and security of models for improved reliability. The DeepBlock-IoTNet secures training operations without employing the individual nodes to perform centralized management.

Baldominos and Saez [7] design the Coin.AI system, which runs a proof-of-useful-work and proof-of-storage schemes in BC networks to respectively verify the trained models and incentivize participants to the verification, and eventually achieve democracy of managing and accessing the intelligence for trust protection. The distributed nodes verify the trained models, and decide the generation and finalization of models using a proof-of-storage consensus algorithm which protects model data and its access.

Weng et al. [131] introduce a DDL framework, DeepChain which supports fair and secure DDL to protect the privacy of models and thus ensure the immutability and trust of data; a value-driven consensus algorithm is designed to incentivize participants to provide reliable training effort, and the training tasks in process are allowed for audit. DeepChain protects the privacy in data and tasks of training participants and supports auditing in the training results.

Sun et al. [114] design an edge-enabled DDL design using two-layers training, where edge devices train models with their data provided by user devices, and subsequently upload their models to CC systems for further training, while BC system is used to store and verify the training event logs to evaluate the trust of participants. Asymmetric encryption is used to protect the parameter sharing between edge nodes and the cloud centralized components, along with BC to protect the trust of authorities.

Zhang et al. [145] propose HP-B which is hybrid parallel algorithm based on BC networks to generate an improved DDL architecture. The design applies grouping
scheme to collaborate the training nodes in BC networks which is determined by the training performance of nodes, their network resource provisioning and training datasets. With BC to verify the grouping decision, both and hence the convergence speed and learning accuracy of DDL are increased to yield improved security and scalability.

Zhu et al. [147] present the proof-of-concept consensus algorithm to solve the security issues in distributed training procedures for federated learning improvement. Using BC technology, this design executes decentralized management to secure the learning coordination and collaboration processes. The proposed consensus algorithm considers the practical Byzantine fault tolerance (PBFT) mechanism and is used to remove the dishonest Byzantine nodes for security improvement.

Due to the feasibility of using BC to support DDL, OD based on BC networks attracts increasing research interest. Typically, OD tasks are large-scale and time-consuming [63]; because the circumstance frequently changes and the uncertain types of objects keep emerging, OD requires large amount of heterogeneous datasets and continuous learning process to guarantee the detection accuracy in front of any dynamics. Correspondingly, storing the data for OD usually considers using the providers which run outside the BC system.

Kumar et al. [53] propose a decentralized OD scheme based on InterPlanetary file system (IPFS) and BC technologies. IPFS is used to store and share the multimedia content in the decentralized peer-to-peer manner. A perceptual Hash approach is designed to convert the content which is uploaded to IPFS, into the unique Hash value which is preserved in BC. The malicious editing on BC causes the inconsistency of content in IPFS, such that the system supports tamper-proof.

Jiang et al. [48] design a novel framework of model sharing for distributed object detection which is applied in vehicular networks. The approach takes BC and MEC networks into account, and enables cross-domain adaptation to improve autonomous driving systems. Edge nodes are employed to train a domain-adaptive YOLOv2 model
with SCs to exchange training parameters and a PBFT consensus algorithm to protect the data sharing.

Shareef et al. [105] introduce a BC-based OD approach which replaces the centralized database with the distributed one using BC technology to enhance the security performance. The designs exploits the distributed storage scheme of BC to facilitate securing network communication, increasing deployment scalability and running real-time services. A YOLOv3 object detection model is trained across nodes, and the IPFS stores relevant data into decentralized applications for distributed training purposes.

Song et al. [110] design a self-positioning correction scheme to improve the accuracy in vehicle position detection. The system requires the multi-traffic signs with a multi-intelligent vehicle positioning error sharing model to be trained by distributed workers, in order to correct the vehicular position information which is originally provided by GPS. In training, information sharing between the participants, such as the intelligent vehicles, common vehicles and roadside units, relies on BC technology for security protection.

Shah et al. [103] propose a BC-based OD scheme using the FL method to eliminate the vulnerability of centralized CA management. IPFS is used as the global storage to save global models and local parameters, which training nodes can exchange local parameters with and retrieve the global model from. The BC platform is enabled to protect the metadata, such as the privacy and model address in IPFS, and the actions to IPFS via SCs.

Islam et al. [45] design a lightweight vehicular BC framework to share models towards DDL with the focus on improving the issues of scalability, efficiency, and transaction verification of existing BC designs. A novel PoVS-BFT consensus algorithm and a two-step transaction verification approach are proposed to enhance the trust and immutability during the distributed vehicular collaboration, and hence to secure the model sharing process.
Hudaya et al. [43] introduce a framework which supports distributed pattern recognition in monitoring events inside IoT-distributed environment. IoT devices are employed into BC networks, such that transactions for event detection by IoT devices can be verified by the BC consensus algorithm. Events from IoT devices are shared over the BC network, and the distributed training using a Graph Neuron (GN) approach is executed to identify the events, such as actions and states of IoT devices.

Ge and Zhou [31] propose FedDetectionBC which is an OD approach based on the training using federated learning method, to address the issues of using central server, especially the single point of failure. BC technology is used to support auditing the distributed training procedures in federated learning to improve the reliability of the entire training process, together with Exchange-FedAvg proposed to improve the training efficiency.

2.4 Security Solution in BC-based VANETs

ZTA is independent of access control models, and much contribution has been made to developing access control models. Typically, role-based access control (RBAC), attribute-based access control (ABAC) and policy-based access control (PBAC) are concluded as the prevalent approaches [116] [32]. Gai et al. propose an access control approach in ZTA based on BC and RBAC which aims to accelerate cross-organizational data sharing. Wang et al. design a dynamic access control, ABEAC, which considers CC as service provider platform and incorporates the user behavior into RBAC for improvement [128]. Dimitrakos et al. design a resource-effective access control model based on ABAC in IoT networks, which extends attribute scope to cover user trust level evaluated via continuous monitoring [24]. Jiang et al. propose CcBAC which is an access control model based on BC and currency tokens and is an improvement of PBAC built into BC for protection IoT applications [47]. Sauwens et al. combine ABAC and PBAC to authorize access from subjects to objects in micro-service system,
and their design supports policies selectivity according to attribute counts and thus ensures dynamic controls [99]. Generally, access control designs based on the prevalent models gain much attention in applying ZTA, and mostly focuses on reliability and the extension of features in making control decision.

Furthermore, since security levels of devices are important for ZTA control policies in VANETs, there are numbers of studies considering device vulnerabilities and potential attacks to improve the accuracy of security assessment. Li et al. propose the MTH-IDS system which evaluates the vehicle and network vulnerabilities, and provides a multi-tiered hybrid IDS approach to detect attacks on vehicular networks [50]. They design a CL-K-means classification algorithm with CAN-intrusion-dataset and CICIDS2017 dataset for security assessment and attack detection. Sedar et al. present a vulnerability assessment of 5G-enabled vehicular systems which focuses on protecting against the category of DoS attacks across multiple network domains [101]. In their consideration, there are a series of DoS attack variants related to various configurations, such that they design a data-driven reinforcement learning approach with VeReMi dataset to detect and identify DoS attacks. Moukahal et al. propose a vulnerability-oriented fuzz (VulFuzz) testing framework which applies grey-box testing designs and uses the vulnerability metrics to identify system security levels [82]. Their security metrics involve the risks of devices, communication, data, control procedures, and even the historical records [81]; they promote grey-box testing method to meet the dynamic network conditions. In general, security assessment is a time-consuming process, and attacks can even be transferred [22]; thus, especially in a busy VANET environment, it is impractical to accomplish an efficient security assessment.

To improve performance, scaling solutions have been remarkably investigated in BC-based VANETs. Most designs follow the divide-and-conquer principle which partitions the BC network into multiple small scopes to accelerate the overall TPS [42]. Wang et al. design a area-based multi-sharding BC framework which aims to jointly preserving privacy and improving scalability in vehicular networks [125]. Similarly,
Zhang et al. focus on improving consensus mechanism with geolocation-based sharding support based on to generate an efficient and trustworthy BC system [144]. For advances, intelligence is injected to sharding scheme. Yang et al. design a clustering-based sharding method which applies K-Means grouping algorithm into the data transaction flow graph (DTFG) to form clusters [139]. They use DTFG to describe the dynamic condition of network, and further use Markov decision process (MDP) and double deep Q network (DDQN) algorithms to optimize clustering parameters to support dynamically arrange shards. Madill et al. design ScaleSFL which is a sharding mechanism together with an extended committee consensus method and applies federated learning to dynamical sharding scheme [76]. In their design, off-chain federated learning is executed for model update verification, which is beneficial for pluggable poisoning mitigation, the shard-level training provides models to be aggregated globally, and a proof-of-concept prototype is designed for consensus purpose. Generally, scaling schemes help improve the dynamic networks, and designating areas followed by monitoring time-related activities for reputation evaluation in areas is effective for shard planning in VANETs.

There are designs of using BC and ZTA jointly to protect vehicular networks which emphasize the improvement on trust model and network scalability. Hao et al. propose an approach using the Shamir secret sharing scheme and a two-layer consortium BC architecture together with a set of SCs to protect the identity and privacy of benign vehicles and the security of their private data. In their ZTA design, zero-sum game theory, contract theory and subjective logic model are used to produce inspection policy [36]. Bhattacharya et al. employ BC to architect a cellular vehicle-to-everything (C-V2X) ecosystem with network function virtualization (NFV) and IPFS to improve the scalability of infrastructure-less topology in three phases to handle data, control and communication. They consider reputation score in defining trust model and use an auction model to arrange policies of resource and service for devices [12]. Wang et al. contribute a design which incorporates two assessment methods on credibility and trust
and utilizes BC to support the collaborative policy management. Data trust is rated based on spatiotemporal-correlated message credibility and dynamic mutual trust; PBFT and proof-of-stake are combined to improve the BC scalability [126]. In these designs, security or trust assessment and performance improvements are necessitated for enabling ZTA in BC-based vehicular networks; but still, the features for policy recommendation are mostly about device activities without sufficient information about security issues and scalability improvement requires extra functions added to the whole chain, which means that the trade-off between security level and scalability persists.

2.5 Research Gaps

As analyzed above, distribution over VANETs can effectively be implemented through CC and BC plus SDN and Virtualization supports for performance improvements. In comparison, centralized management facilitates the distribution achievement, but meanwhile exposes the chances of attacking the system; while BC spreads the authority over multiple parties and requires consensus to protect the trust of data, such that the distribution constructed based on BC can boost the reliability. However, it is impractical to transfer large volume of data inside BC transactions which are constrained by size and time quota, and multimedia data for DDL usually exceeds the capacity of transaction, and hence tend to encounter data loss during transmission. The immutability of BC has attracted extensive interest in its integration with VANETs for traffic information protection, but the related applications rarely clarify the design of traffic data for being compatible with BC data requirements. Besides, most BC-based applications in VANETs evolve around protecting the traffic data with authentication and permission, but rarely consider the data attributes such as type, size and structure in transmission.

In terms of DDL designs, basically DDL involves the representation, transfer and
interpretation of models between training candidates, and these procedures can be promoted by using the centralized management and the global storage. Collaboration with centralized management counts on the information collection and analysis in the central components, but the exposure of central components increases the risks of being attacks. Restricted condition of VANETs and managing components would also cause unfair connection and network blocking which bring about the overhead and latency of cooperation adjustment and hence increases the staleness. Vector-based map-reduce and grouping schemes are able to accelerate the collaboration, but algorithms for defining vectors and groups are complex and designed in center points. Moreover, share memory mechanism becomes popular to speed up the parameter sharing; however, it is mapped between center servers and training workers. Generally, centralized management helps with creating new DDL designs, because the major consideration is narrowed down to a small area; but for the same principle, sensitive data and key functions are saved into the center points, which makes it easier for attackers to control the system by hacking the center points.

BC technology is competent at solving the centralization risks, as it spreads the management functions over multiple parties and meanwhile to conceal their privacy and sensitive data to avoid malicious intrusion. But there are a series of limitations in BC, firstly transaction capability is limited because it needs to be verified timely, secondly block size impacts the consensus and verification latency, data synchronization causes burdensome communication in BC networks, and so on. To reduce the data size in communication, most designs focus on using external storage, such as IPFS and CC system, to preserve the real content, and using BC to save the context and metadata to reduce the communication overhead; but the interaction is rarely explained in detail. For example, most designs lack the description of SCs which they use for interaction in dataset retrieval and model sharing. However, the execution of SCs directly impacts the DDL performance. Moreover, consensus algorithms are the major components to be improved in BC-based DDL, but mostly consider the superficial
elements, such as the signature and hash verification. Actually, the DDL process incorporates data and tasks, therefore consensus should be performed in two directions to provide comprehensive protection.

The related works in protecting BC-based VANETs demonstrate the significance of BC and ZTA in protecting VANETs, but they meanwhile reflect the gaps to be filled for further improving system security. Prevalent access control models can be further improved for better performance in VANETs. AI efficiency in ZTA and BC scaling schemes should consider the mobility and capability. The features for AI training should involve the cybersecurity elements, such as reachable infection caused by the unhealthy devices, and protection adapted to network status, since these factors eventually impact the privacy and trust protections. The decentralized protection design of BC sacrifices certain performance and unavoidably imposes overload in VANETs, and reliable ZTA control requires to sufficiently consider cybersecurity elements of devices. Thus, it is demanding to improve the scalability of BC-based VANETs to meet the complex cybersecurity requirements, and consensus and validation algorithms need to support the efficient interoperability between main-chain and shards for cross-shards data sharing. Overall, the synthesis of ZTA, BC and VANET technologies still is concerned with the BC trilemma issue.

In summary, there are four major points that can be improved in designing our proposed system. The BC functional expansion and internal structure can be enhanced in terms of processing various types of data, the management of sharing DDL data, including inputs, parameters and updates, can be decentralized for better protection with BC mechanism, BC-based DDL should not only consider the data but also the task distribution approaches for performance improvement, and the adaptive security solutions on BC-based DDL are required for efficiency and reliability in VANETs and can be improved by advanced algorithms and extended features.
Chapter 3: Proposed Design

3.1 Overview

The proposal focuses on addressing the issues of reliability and efficiency of DDL processed in VANETs, so as to improve the security in developing road-traffic-related intelligent applications, such as autonomous driving, intelligent transportation system, railway control, etc. The proposed design is illustrated in Fig. 3.1, where VANET nodes are equipped with virtual network interfaces and BC functionality for distributed communication, and DDL as based on BC distribution is a progressive training process alongside the block propagation.

![Diagram of proposed vDLT-based DDL in VANETs.](image)

Figure 3.1: The proposed vDLT-based DDL in VANETs.

As shown in Fig. 3.1, BC network is deployed based on vDLT platform, which allows virtualized resource for communication, computation and storage. Besides the fixed servers being vDLT participants, VANET nodes, such as vehicles, traffic lights, RSUs and street lamps, run vDLT system to interact with the BC network,
such that they can share the visual information and the AI models to execute OD. Transactions defined by SCs are utilized in this interaction and in the distribution of tasks and datasets for training. Heterogeneous devices can form various training sites to train model in the distributed manner, and the fixed servers are committed to handling the model aggregation, compression and trust consensus during block processing. Macro base stations (MBSs) and small base stations (SBSs) are employed to support extensive and ultra-rapid communication between the VANET nodes in a large scale.

In the proposed architecture, the reliability can be basically ensured by the overlay distribution and the BC fault-tolerance which jointly overcome the unsteady communication among VANET nodes; besides, to avoid the central management or global storage, asynchronous parameter updates can be performed alongside the block delivery and consensus across BC participants; training tasks which are distributed over heterogeneous VANET nodes, will generate relevant parameters, and their relation is verified to measure the DDL reliability. In terms of efficiency, progressive model aggregation can support OD-while-training, in which the basic knowledge is rapidly generated for OD and subsequently the knowledge is continuously accumulated to improved OD accuracy; meanwhile, underlying communication can take the advantage of emerging mobile network technology, especially the 6G design which goes beyond 5G networks and supports higher-level uRLLC for both reliability and efficiency improvements; ZTA access model can be integrated into BC participants to protect data sharing for secured DDL, and the model can take device security status as the primary attribute and extra attributes, such as of role and data features, to adapt to network variation; sharding scheme can also be enabled for further efficiency improvement considering the security indexing and traffic statistic, and supports cross-shards management together with access control and cooperative consensus algorithms.
3.2 Research Questions

According to the proposed design, there are three major domains involved, which are respectively the communication scheme of VANETs, the distributed ledger mechanism of BC networks and the integration of DDL into the BC distributed architecture. For example, the intensive data communication required in DDL and BC networks are believed to have conflicts with VANET limitations; the existence of trustless nodes in BC networks may increase the risks of comprising the system integrity and hence the quality of trained models; without the central management or global storage, the collaboration of DDL is constrained by the performance of data synchronization over the BC network. Therefore, to investigate the significance of proposal and the feasibility in developing the proposed system, there are a series of questions need to be analyzed.

- How the efficient communication is supported in BC-based VANETs to highlight the advantages of running VANET nodes in BC networks?

- What information is needed to share between VANET nodes in the BC network for intelligent application designs, and can the traditional BC design support this data sharing?

- How can the BC mechanism be improved to protect the information sharing at higher efficiency and better reliability to support the distributed data-intensive application?

- How are the DDL tasks coordinated by the BC participants without the central controller, and how can the decentralized management of BC be trusted?

- How is the security of BC-based VANETs protected since the mobility and self-governance issues in VANET and BC systems challenge the traditional security solutions?
• How can the performance be improved confronting the conflict between resource consumption in BC consensus and the resource limitation in VANET architecture?

### 3.3 Contributions and Publications

#### 3.3.1 Contributions

The proposed research is executed in four stages. Each stage is focused on addressing the issues in a specific domain, and the contribution from each stage is progressively accumulated towards the eventual achievement of the proposed solution.

In stage one, the vDLT system is developed to support real-time multimedia networking and evaluated in terms of the performance with the focus on sharing surveillance videos; with the capability of vDLT in transmitting multimedia data, the vDLT-based VANET network is designed and evaluated in comparison with routing protocols. VANETs are constructed based on vDLT system which establishes an overlay network for maintaining sustainable distribution between VANET nodes regardless of the routing paths frequently changing in the physical plane, such that the communication achieves remarkable reliability. The contributions are listed as follows.

- Virtualization of network resource is developed to vDLT, and constructs a overlay network for VANET scenarios. The performance is evaluated by simulating the vDLT-based VANET, and results approve the reliability gains in the unstable wireless networks.

- Multimedia streaming is developed into vDLT as well. With this feature, real-time surveillance networks and VANETs are built in vDLT system to respectively capture the surrounding and traffic events, and both achieve positive
performances.

• An algorithm is developed in vDLT to generate frame fingerprints using real-time extraction of packets from surveillance multimedia streams, SHA3-512 functions and the verification of frame consistency.

• Consensus of the frame fingerprints is designed into vDLT, and transcoding and preservation of multimedia streams is distributed in vDLT which eventually ensures recordings tamper-proof.

In stage two, vDLT is improved to protect the sharing of traffic information in VANETs, based on the contribution from stage one. The innovation of vDLT involves the layered design for data interaction inside system, the trust management on traffic information in the higher level of granularity, and a set of advanced system functions and SCs. Generally, traffic information is processed in video frames which are wrapped into blocks as transactions for validation and consensus; inline SCs automate the interactive verification of traffic frames without intervention. The contributions are listed as follows.

• Visual traffic information is processed at the level of frames for higher accuracy, and frame verification and digital watermarking is developed for stronger protection. A set of collaborative SCs are developed to exchange traffic frames with BC and an algorithm to verify the authenticity of frames according to their multimedia context.

• A 3-tires model is designed to integrate BC and VANETs. Efficient multimedia streaming, agile access points to BC, and reliable data protection with BC storages are developed in three layers to share traffic information via BC in VANETs. The features of multimedia frames and BC transactions are exploited to enhance their cooperation and the validation of traffic data.
• Our system is able to extract the sensitive data in frames and dynamically adjust the number of frames wrapped into a transaction to solve the limitation of BC, such as limited transaction size and processing time. According to the metadata, such as the types and sizes of frames, the system can construct frame transactions in higher storage efficiency, and meanwhile reduce the latency of sharing and boost the system robustness.

In stage three, DDL is integrated into vDLT, and the system is evaluated as compared with CC-based DDL. Since traffic information sharing and storage can be secured by vDLT system in VANETs, it is reasonable to enable DDL in vDLT in order to support the detection of traffic events. In this design, DDL is integrated into vDLT networks where the collaboration of training is managed by vDLT participants, the system reliability is enhanced by protection in both task and model directions, and to in-depth investigate the system strength towards future network technology, the 6G network scheme is applied. The contributions are listed as follows.

• A improved DDL system based on vDLT and 6G networks is proposed. This vDLT-based DDL takes the advantage of uRLLC in 6G networks to explore the BC-based DDL potential, together with a weight-based parallelism to arrange training tasks over edge devices at different resource levels.

• Models are generated at MBS and SBS levels, and a two-tiers aggregation scheme is designed for progressive model production, where SBSs and MBSs respectively work in the synchronous and asynchronous collaboration manners; a dual-driven consensus algorithm verifies the tasks and parameters by exploiting the transaction rationale, rather than merely checking the block metadata.

• A set of small-granularity SCs in vDLT are developed for regulating parameters, training tasks and model requests; parameter and task SCs are inline connected, such as for updating task records via parameter sharing and vice versa; users
can retrieve CNN models through SCs for object detection.

In stage four, vDLT-based DDL will be further improved by adding the supports of ZTA access controls and sharding schemes. Previous stages demonstrate the feasibility and advantages of DDL in VANETs based on vDLT, and the process is accompanied by the support of intensive data sharing. Thus, it is necessary to explore the state-of-art security solutions to protect the data so as to improve the reliability of related DDL applications. Meanwhile, the fusion of security and consensus solutions into VANETs for protection sacrifices system performance and unavoidably imposes overload in VANETs, such that it is demanding to improve the system scalability to achieve efficient protection. The contributions are listed as follows.

• We propose a framework which highly synthesizes ZTA, BC and VANET elements in protecting data sharing, and the framework can support top-down management for efficient and reliable protection. Cooperation of consensus procedures, and coordination support are designed into vDLT for cross-shards validation and consensus.

• The system components and dataset features are designed along with the supportive algorithms regarding security assessment, hierarchical access control and sharding scheme being presented in detail. Cybersecurity issues consider the device reachability as risk infection opportunity and sharding schemes consider both traffic and security situations.

• The development of framework is accomplished, and the system performance are analyzed in experiments which perform evaluation and result analysis to demonstrate the system significance. Proposed hdPBAC is demonstrated swift in response to varying communication patterns.
3.3.2 Publications


Chapter 4: Multimedia vDLT and vDLT-based VANETs

There are two tasks in this stage where one is developing the multimedia networking supports into vDLT BC and the other constructing VANETs based on vDLT for reliable traffic networking system. The multimedia support is implemented in a project which requires to transmit and protect surveillance information through BC. Surveillance information incorporates the multimedia of video, audio and text messages. The support for this real-time surveillance can be smoothly moved to VANETs for sharing traffic events. Moreover, with the project contribution, a vDLT-based VANET framework is designed to evaluate the capability of vDLT in improving the communication performance of VANETs. The framework innovates traditional VANET networking schemes by applying overlay networks and distributed networking functionality into the VANET devices. Evaluation results support the potential of vDLT in terms of protecting the information sharing in VANETs.

4.1 Multimedia Networking via vDLT

Surveillance is being pervasively used, and its recording is extensively applied in practice. Thereby, protection of surveillance recordings, which is related to multimedia security, increasingly attracts research interests. Traditional techniques, such as watermarking, cryptography and steganography, focus on analyzing multimedia content, resulting in high complexity and long latency, which makes them not suitable for protecting real-time surveillance applications. In this paper, we propose a novel BC-assisted framework to protect the recordings in real-time surveillance applications. In the proposed framework, we design an algorithm that generates frame fingerprints, which involves real-time extraction of packets from surveillance multimedia streams,
SHA3-512 functions, and the verification of frame consistency. We use a BC for preserving the frame fingerprints generated by our proposed algorithm. Different from existing works that use simulations to show the performance, we develop a real system using a service-oriented BC which employs virtualized distributed ledger technology, and demonstrate the effectiveness of proposed system.

4.1.1 System Design

4.1.1.1 Architecture

With the background revealed, we propose a system which employs BC technology to protect the frame information, and accordingly prevents the surveillance recording from being tampered with. Generally, our solution focuses on the development of BC for preserving the frame fingerprints which are generated by our proposed algorithm, and
additionally creates cloud computing services to consolidate the storage of surveillance content.

The architecture of the proposed system is depicted in Fig. 4.1, which incorporates three major physical scopes, i.e., virtualized distributed ledger technology (vDLT) as a BC, cloud computing platform and IP cameras. According to Fig. 4.1, cameras use multicast and RTP components to send video stream to multiple recipients, while both vDLT and cloud computing suites arrange nodes to join the same multicast group to receive video stream.

**Multicast and RTP** Multicast, which is commonly based on UDP and supporting one-to-many communication, allows to send streams in the distributed manner. It can guarantee the synchronization of same data amongst the clients of the same multicast group, and can accordingly mitigate the risk caused by unitary receiving point effectively. Besides, RTP which runs over UDP, provides necessary control during transmission by using RTCP. With RTP for streaming, information, such as the source ID and timestamp is exchanged between counterparts to prevent data loss. As compared to TCP which requires permanent connection and frequent retransmission, RTP can achieve higher efficiency of performance and resource consumption with the benefit from UDP. Therefore, the combination of multicast and RTP is used in this project to comprehensively improve the streaming quality.

**Cloud Environment** The CC platform acts as recipients of surveillance streams, and saves the information of frame identity which is mapped with metadata in vDLT for frame verification. In practice, a cloud can be constructed using a number of mature options, such as Google Cloud, Amazon AWS, OpenStack and Kubernetes. In the cloud environment, a number of video receivers, video processors, and a clustering storage are equipped and managed in a centralized way. Video receivers are placed at the edge of the platform to interact with IP cameras and video processors. Procedures
in video receivers exchanges RTCP with cameras and feeds streams to the storage in cloud. Correspondingly, video processors read streams from the storage and process them in decoding, encoding and validation functions. For higher efficiency, video processors are scheduled by a load balancer. Eventually, the output from video processors is fed back to the storage for conservation. In addition, video processors are the interface interacting with HTTP services in vDLT. In this cloud, the load balancing and clustering storage techniques help improve the performance and security on manipulating video streams.

**BC Platform** vDLT is introduced to provide the functionality of BC in our system. vDLT is an experimental platform which constructs a virtualized environment to enable service-oriented BC technology with DPoS consensus, and applies consensus into managing virtualized resources to increase the robustness and performance [141]. As a receiver of surveillance streams, vDLT employs participants equipped with multimedia data sink (MDS) service to interact with IP cameras. Besides handling RTCP, these participants also extract RTP content and context, such as packet sequence, timestamp, and source IP address and port, followed by generating the fingerprint of relevant frames. By using SCs, MDS nodes create transactions to carry a bundle of fingerprints, and broadcast the transactions over the vDLT network. After receiving those transactions, a DPoS-scheduled producer in vDLT creates blocks to wrap them up, and eventually broadcasts the blocks for distributed validation, consensus and preservation. In addition, vDLT exposes HTTP service to the video processor in cloud for the verification of frame fingerprints. Due to the synchronization and consensus between multiple nodes in vDLT, the fingerprint of frames are solidly protected and unalterable. Moreover, the frame rate commonly ranges from 30 FPS to 120 FPS, while DPoS is proved to excel at processing more than thousands of transactions per second (TPS) owing to the effort from multiple producers, therefore vDLT is efficient at processing surveillance frames. In general, the service-oriented
scheme in vDLT facilitates the expansion of functionality; protection on frames even just involving random individuals is more scrupulous, coupled with immutability of vDLT to further protect the trust of surveillance recording.

4.1.1.2 Proposed Algorithm

The algorithm majorly proposed is illustrated in Algorithm 1, in which the critical functions of system are executed in stages to support multimedia streaming. Moreover, SCs, RTP specification and transcoding configuration are the necessary parameters involved in process. According to Algorithm 1, there are four stages running system initialization, stream reception, transcoding and fingerprint processing.
Algorithm 1: The major algorithms proposed to support system.

**Input:** MG address, MG port, MDS SC

**Output:** SDP file, MDS account, frames

**Initialization:**
Designate Multicast Group (MG) Address & Port
Generate Session Description (SDP) file
Distribute SDP file to vDLT MDS service and video Receiver
Open MDS & Camera accounts in vDLT
Push MDS SCs to vDLT

**Receiving Frames:**
Read SDP file → Join MG → Extract RTP context

while true do
    Retrieve RTP packet $N$
    Verify RTP timestamp & sequence $N_s \leftarrow \delta T_s = T(N_s) - T(N_{s-1})$
    Parse packets $\{N_s, N_{s-1}, \ldots, N_{s-fps+1}\}$ to produce frame $F$
end

**Transcoding Frames:**
Read RTP context → codec configuration

while read frames do
    Filter frame $F$ → filter results cached
    Encode frame $F$ → mp4 ← codec & filters
    Save RTP context & frame $F$ & mp4 → vDLT ledger
end

**Generating Fingerprint:**
Retrieve SC in vDLT → $SC_{mds}$ ← MDS account

while read frames do
    Verify timestamp $Ts$ of frame $F \leftarrow 1/fps = \delta Ts_F$
    $SHA3 - 512(RTP\ context, F\ content, timestamp) \rightarrow \text{fingerprint } F_{fgp}$
    Build transaction $TRX \leftarrow F_{fgp}, F$
    Push $TRX \rightarrow ID_{TRX}$
    Sync. $TRX$ across vDLT
end
**Initial Stage**  To synchronize the streams, a session description protocol (SDP) file is generated and distributed to all the surveillance senders and receivers. The SDP file mainly contains the information about SSRC, stream format, sampling frequency, IP address, port, and video meta-data. RTP components follow the SDP file to initialize and maintain the communication. To enable multicast, a multicast IP address should be set ahead, and accordingly the participants using the same SDP file will join the same multicast group. To interact with BC, the accounts of receivers and senders, which respectively represent MDS service and camera, as well as the corresponding SCs, need to be opened in vDLT.

**Receiving Stage**  Based on the SDP file, subscribers join the dedicated multicast group to receive streams from cameras. Before looping to retrieve payload packets, the receiver establishes connection with sender, from which the RTP context is extracted. In order to obtain a frame, the receiver caches multiple RTP packets and converts those carrying the same RTP timestamp to an entire raw frame. In this stage, high efficiency is guaranteed, since the raw frame is un-formatted and of small size, especially those of forward-predicted (P) frame and one bi-directionally predicted (B) frame.

**Transcoding Stage**  Transcoding service is necessary in cloud, for the purpose of re-formating and visualizing the streams received from surveillance cameras. Transcoding consists of decoding and encoding, in which decoding uncompresses the raw frame to produce the picture of source format, such as 640x480 image size and H.264 codec, while encoding compresses the decoded frame by using a specific format, such as MP4 and Flv. To enhance the reliability and efficiency, transcoding services are enabled redundantly, and scheduled by a load balancer based on the status and usage of resource. Eventually, transcoding services save their output and the input from Receiving Stage in cloud, to create the mapping relationship.
**Fingerprint Processing Stage**  As parallel to transcoding, processing fingerprint is handled in vDLT. The fingerprint pertains to the content of raw frame without running the decoding procedure. The equations below are used to estimate the performance in this stage.

\[
T_{inc} = \frac{S_{feq}}{F_{rate}} \quad (4.1)
\]

\[
T_{trx} = \sum_{1}^{n} (T_{dmx} + T_{gen} + T_{c} + \Delta T_{dyn}) \quad (4.2)
\]

\[
R_{fgp} = \sum_{1}^{n} (N_{trx} \times B_{fgp}) \quad (4.3)
\]

\[
= \left\lfloor \frac{1}{T_{trx} + T_{smt}} \right\rfloor \times B_{fgp} \quad (4.4)
\]

- Equation (4.1) calculates the fixed timestamp increment, in which \(S_{feq}\) and \(F_{rate}\) respectively are sampling frequency and frame rate;

- Equation (4.2) indicates the time for building a transaction of \(n\) fingerprints. \(T_{dmx}\) is the time for extracting frame packets, \(T_{gen}\) for fingerprint generation, \(T_{c}\) for reading parameters of system, and \(\Delta T_{dyn}\) for dynamic delay between frame arrivals;

- Equation (4.3) and (4.4) can estimate the rate of writing fingerprints into BC per second. \(N_{trx}\) in (4.3) stands for the number of transactions in a block, and \(B_{fgp}\) for the batch size of fingerprints in a transaction. In (4.4), \(T_{smt}\) is for the processing time inside SCs, and affects the block production of current transaction.

Furthermore, the following explains a series of major task executions in this stage.

- SHA3-512, which produces 64-bytes digests, is the core function for fingerprint generation, with the arguments of RTP context and raw frame content. Since the smaller hash spaces increases the risk of collision, SHA3-512 is more capable
of protecting tremendous surveillance streams against collision attacks than its predecessors, such as SHA3-256, SHA2 and SHA1.

- Before generation, the sequence of frames should be verified based on the timestamp increment, and those intermittent frames will be marked down. Subsequently, SCs from Initial Stage are used to construct the transactions, which hold the fingerprint and its context.

- The MDS SCs are exposed to the service in vDLT, while the Camera SCs as an inline one is concealed and called only by MDS SCs. Inversely, the retrieval of fingerprints from BC is routed by MDS SCs. Generally, MDS acts as an agent of processing fingerprint in vDLT. This design can help prevent the information about the camera and fingerprint from being stolen through hacking MDS service.

- Moreover, transactions containing fingerprints are broadcast over the vDLT network, and they are wrapped to blocks by the producer nodes. It is of higher security to disperse the block-consensus and fingerprint-processing functions over different participants and hence to mitigate the intensive attacks.

In this proposal, vDLT providing BC functionality is the critical element to accomplish the comprehensive protection on the frame fingerprint. The DPoS consensus algorithm is efficient and vigorous to impede the malicious modification to the fingerprint. The utilization of cloud facilitates, rapid transcoding, and secured storage helps to avoid data loss. Any tampering attacks on the surveillance recording in cloud will lead to the inconsistent fingerprint generation compared with vDLT, and thus jeopardizes the trust of recordings. As aforementioned, the trustworthy surveillance recording is imperative for the forensic and IoT application; correspondingly our proposal can prevent the recording from being tampered with and hence meet the purpose.
4.1.2 System Development

4.1.2.1 Multimedia Streaming Services

The development of our system focuses on the services in vDLT and cloud environment. Streaming function in IP cameras is provided by the tool of FFmpeg [8]. Besides, the APIs of FFmpeg are invoked to implement the video receiving and transcoding components.

The design for multimedia streaming is illustrated in Fig. 4.2, in which Video Receiver and Processor are the two major components to be developed. According to Fig. 4.2, there is a MongoDB used for data persistence, which can be consistently manipulated by both Video Receiver and Video Processor; such that, MongoDB serves...
as the mediator and communication media between the two Video operators. The designs of Video Receiver and Processor are explained below.

**Video Receiver** is built with WebSocket to allow external control.

- Clients can send HTTP request of START (or STOP) with SDP file to start (or stop) Video Receiver thread.
- For START, this thread allocates resource based on SDP content, and loops to receive streams from a camera and then to send them with codec and RTP context to MongoDB.
- For STOP, WebSocket daemon kills the thread and withdraw resources.

**Video Processor** supports a group of threads, and also external control with WebSocket daemon.

- Clients can send HTTP request of START, STOP or SEARCH. START or STOP is used to control the boot-up or shut-down of threads, while SEARCH is used to retrieve the fingerprint of frames.
- For START, three threads and two queues are created.
  - One thread reads records of codec, RTP and frames from MongoDB, and then writes them to the first queue;
  - The second thread decodes and filters the items in the first queue, and writes to the second queue;
  - The final thread processes encoding after reading the second queue, and writes back to MongoDB.
- For STOP, WebSocket daemon can kill individual threads and withdraw the related queues and resources according to the arguments.
For SEARCH, WebSocket daemon takes the information, such as the timestamp, sequence, source IP, etc, from HTTP request, and proceeds to:

- retrieve the records from MongoDB based on the request.
- invoke CURL APIs to query the mapping fingerprint in vDLT.
- compare the fingerprints in both sides.

4.1.2.2 Multimedia Data Sink Services

MDS service receives streams from cameras, utilizes SCs to interact with the internal BC storage, and employs HTTP service to communicate with the external clients. Fig. 4.3 depicts the components and their arrangement to support MDS service in vDTL. As shown in Fig. 4.3, MDS exposes three interfaces to HTTP service, which
respectively handles the boot-up and shut-down of PacketReceiver threads and the query of fingerprint from vDLT chain. SCs for Agent is called to push the transactions of fingerprints, which are processed and uploaded to the ledger by SCs for Camera.

- **MDS Main** thread retrieves the Agent SC description according to the configured account information, and creates a shared message queue, i.e., AVTMQueue to communicate with its sub-threads. Subsequently, it starts to listen to the request of HTTP service and to monitor the AVTMQueue.
  
  - Upon receiving boot-up request, the thread parses request for SDP content, and creates a group of PacketReceiver sub-threads based on the stream information in SDP, coupled with saving the group ID.
  
  - Continuously, the main thread monitors AVTMQueue, and once the new items have been pushed, it will be notified to call UPLOAD functions to process fingerprint transactions.
  
  - For shut-down, the groups of sub-threads and resources will be released by the main thread, and the AVTMQueue will be flushed to handle the remaining frames.
  
  - For search request, the thread invokes Fingerprint-Retrieval component with the request as parameters, and waits for the result followed by returning it to HTTP service.

- **PacketReceiver** sub-threads threads reuse the receiving functions of Video Receiver in cloud, and adds the generation of fingerprint next to the extraction of raw frames.

  - Upon starting, sub-threads join the multicast group based on SDP to subscribe streams from cameras, and write the RTP context and fingerprints into AVTMQueue.
While being shutdown, sub-threads stop reading streams from FFmpeg container. Next, they flush the FFmpeg container to handle the remaining streams and close it to leave the multicast group.

- **UPLOAD** functions are called by the main thread to bundle and upload fingerprints and RTP context in transactions.
  - By calling chain-related services, UPLOAD functions obtain the chain ID and reference address, and cache them in a global memory.
  - With the Agent SCs description and the chain information, transactions can be created to wrap up the frame data.
  - The private key of Agent, which is created in advance, is used to sign the transaction. On the other hand, its public key can be retrieved from chain to verify the signature.

- **Agent SCs** works as a lightweight proxy. Since the de facto payer is camera, there is no storage to be consumed by Agent SCs.
  - In Agent SCs, the action basically is to forward the data to Camera SCs without saving any data.
  - Data fields include Unique Header generated by hashing the current time and camera information, the account name of this MDS service, RTP context about the multicast IP, and frame information on sequence, timestamp, fingerprints, sizes, and optionally its content;

- **Camera SCs** manipulate the stream data at the backend. The storage of streams in this contract encompasses Header table and Content table. This contract belongs to the account of camera, who accounts for the expense on resource consumption in vDLT.
- Actions in this contract support Increase, Remove and Query. The action name is forwarded by Agent.

- Data fields are consistent with those in Agent SCs, but the data from Agent is parsed to the storage structure in Camera SCs.

- Header table and Content table conform to one-to-many pattern. Unique Header is the primary key for Header table, while a self-generated hash value for Content table. Besides, Content table is indexed by sequence, timestamp, and IP address.

- **Fingerprint-Retrieval** component allows query to chain data from HTTP service. Simply, without SCs, this component accesses the chain database via account information. Therefore, no expense is required while using this component to query fingerprint.

  - The request provides query conditions, such as value related to primary key or index in tables.

  - The component selects and returns the records to HTTP service, which later forward the result to clients.

In summary, our development takes the advantages of threading and message queue to increase the throughput as well as the efficiency. MDS service relies on vDLT, and exploits the relationship of SCs to enhance the security of the data. With the consideration of economic aspects, storage is created for camera accounts which are the actual consumer other than the account of MDS service. Furthermore, the development of service in cloud is generalized, which means that the service can be run in a standalone or cloud computing environment.
4.1.3 Evaluation

The emphasis of evaluation is put on the performance of MDS service in vDLT, since it remarkably affects verification of fingerprint. Besides, evaluation includes the feasibility of using fingerprints to detect the tampered frames in cloud.

4.1.3.1 Environment Preparation

To setup the environment for evaluation, a list of software is needed to install ahead of time, which is recorded in Table 4.1. Besides, the parameters for experiment are in Table 4.2.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boost</td>
<td>1.67.0</td>
<td>extension to C++ standard library</td>
</tr>
<tr>
<td>FFmpeg</td>
<td>4.2</td>
<td>multimedia tools and APIs for development</td>
</tr>
<tr>
<td>MongoDB</td>
<td>3.5</td>
<td>clustering storage and drivers (C/C++) to access storage</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>1.17</td>
<td>platform of containerization cloud</td>
</tr>
<tr>
<td>Weave Net</td>
<td>2.6</td>
<td>construction of overlay network</td>
</tr>
<tr>
<td>Ingress-Nginx</td>
<td>0.26</td>
<td>controller of load balancing</td>
</tr>
<tr>
<td>vDLT</td>
<td>alpha 0.1</td>
<td>platform of BC</td>
</tr>
</tbody>
</table>

Table 4.1: Tools used to construct the project.
Table 4.2: Specifications for system evaluation.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Num of Nodes</th>
<th>Specification for each</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP-Camera</td>
<td>1</td>
<td>CPU: 3.2 GHz(4 cores), RAM: 4GB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GoP: 250, FPS: 30, Bit-Rate: 512 kb/s</td>
</tr>
<tr>
<td>Kubernetes</td>
<td>2</td>
<td>CPU: 3.2 GHz(4 cores), RAM: 32GB, Storage: 1TB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1×Pods: receiving service, 2×Pods: processing service</td>
</tr>
<tr>
<td>vDLT Network</td>
<td>3</td>
<td>CPU: 3.2 GHz(4 cores), RAM: 32GB, Storage: 1TB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Producers: 3, Block-Time: 0.5s, Fingerprint-Batch: 10</td>
</tr>
</tbody>
</table>

As FFmpeg provides the compiled tools for video capturing and streaming, the computers with IP cameras installed can execute FFmpeg tools to serve as a surveillance media. The camera in our system utilizes H.264 for coding, RTP for transmission and yuv420p as pixel format. Other parameters, such as bit rate, image size and GOP size, are recommended by H.264 to use common cameras.

For cloud computing, Kubernetes [41] and Weave Net [130] are integrated to accomplish a cloud of containers, plus using Ingress [52] for load balancing and MongoDB [80] for the clustering storage. Weave Net working in Linux kernel space constructs a L2-based overlay network for speedy cross-hosts and multicast communication. Simply, 2 replications of Pods are designated for Video Processors and load balancing, while 1 Pod for Video Receiver.

In vDLT, three producers participate DPoS consensus, and the time slot for block production is set to 500 milliseconds. One transaction carries up to 10 fingerprints. Producer nodes are connected using the ring topology. The MDS service is installed in one vDLT node which is connected with the node running HTTP service and
one producer node. In addition, because Boost [9] C++ libraries are used for our development, they are required by the services in both vDLT and Kubernetes in run-time.

4.1.3.2 Evaluation Results

The main purpose of evaluation is to check whether the tampered frame holds a different fingerprint than the original one, and whether the frame fingerprints can promptly be received and pushed into blocks without any loss.

The demonstration of detecting tampered frames is shown in Fig. 4.4. We replace the left picture with the right one in cloud system to mimic the tampering attack. From Fig. 4.4, it is seen that the left picture contains a vehicle which is missed on the right side, and these two different frames generate unequal fingerprints. This can be related to a real scenario in which suspects fake the proof of absence from criminals scene by removing their vehicles from the left frame to generate the right frame. However, the judiciary can refer to the correct fingerprint which is associated with the left picture and unalterably conserved in BC to reject the forged evidence.

Throughput is another performance metric in our evaluation which is shown in Fig. 4.5. Note that the pixel format of YUV420P and the compression of H.264 determine the light-weight packets in transmission, therefore, rather than to test the network performance, the throughput here is used to reflect the reliability of our system.
GoP size is set to 250 and FPS to 30, it is consistent that the peak of throughput, which is caused by I-frames transmission, occurs approximately every 8 seconds, as seen from Fig. 4.5. Handling B-frames and P-frames is seen at the lower level of throughput. The highest value at the beginning is contributed from the retrieval and analysis of stream context.

Furthermore, the time efficiency is evaluated through comparing the time consumption in the major procedures. Time sequence and delay are respectively compared in Fig. 4.6 and Fig. 4.7. According to Fig. 4.6 and Fig. 4.6 together, the procedures which include receiving frames, pushing frames into transactions and wrapping transactions into new blocks proceed steadily except in the beginning stage. For the first 200 frames, the initialization of system, such as preparing SCs description, chain ID and stream context results the lowest efficiency of being packed into blocks. Besides, in the beginning stage, redundant packets are processed until the stream context is clarified, such as decode context and timestamp, which is necessary for maintaining the streaming quality. In general, the time spent on creating fingerprint transactions is stable and merely hovers around tens of milliseconds. Since a batch of frames are processed into one transaction, the time distance between frames is included; in other
Figure 4.6: Comparison of time sequence in different producers.

Figure 4.7: Comparison of time latency in different producers.
words, high efficiency is witnessed in our system.

The last evaluation pertains to the efficiency of wrapping frame fingerprints into blocks, which is shown in Fig. 4.8. At the beginning, numbers of redundant packets are cached in system, and once the initialization is finished, they are immediately pushed into transactions and then into blocks. Therefore, in Fig. 4.8, the earlier blocks are witnessed containing more frame fingerprints. As the system get well-prepared, the performance of wrapping fingerprints into blocks becomes steady, and it is clear that around 30 fingerprints are obtained by each block. The results show that the system is prompt and reliable to create blocks for protecting frame fingerprints.

Overall, the evaluation demonstrates that our system is feasible and efficient to protect the surveillance recordings. Fingerprints are generated promptly without any loss, so that they can be entirely uploaded to BC so as to enforce the solid protection on surveillance recordings. On the other side, the evaluation shows that services in cloud can properly receive and transcode the recordings to support the succeeding verification of fingerprints.
4.1.4 Remarks

Inspired by the immutability of BC, and considering the importance of surveillance recordings, we developed a system to protect the fingerprint of frames via BC technology. In this paper, vDLT, which is a service-based BC platform, is introduced, and our proposal focuses on the development of MDS service and two SCs in vDLT to process the fingerprint of frames. MDS service is engaged in generating fingerprints for the received frames and using its SCs to push the fingerprint information into vDLT. To secure the access to vDLT storage, the two SCs are designed in an agent-to-sink pattern, in which the MDS SCs are exposed as an agent while Camera SCs is concealed as a sink, and frame fingerprints are forwarded by agent to the sink for their preservation. As without complex economic regulation, our design considers cameras as the actual consumer of storage, since they are the source of frames while MDS service serve as a courier coupled with frame processing. The performance evaluation shows that our system is efficient and reliable in terms of processing the frames and preserving their fingerprint in BC. Moreover, upon editing the content of frames, the result of fingerprint verification proves the feasibility of our system. In addition, to support a secured storage of surveillance recording, cloud computing is enabled in this project, and which is established by using Kubernetes along with a series of plugins. The services in this cloud focuses on transcoding the surveillance streams and conserving the recording for latter verification, and the efficiency and security are improved by using the load balancing scheme and clustering storage. In general, our project incorporates BC technology and cloud computing to protect surveillance recordings, and the protection is concrete and consolidated by delving deeply into the frame level.
4.2 Distributed VANETs based on vDLT

Recent advances of routing algorithms have greatly improved the reliability and efficiency of vehicular ad hoc networks (VANETs). But the constraints of network resources result in a trade-off between reliable data transmission and the performance of routing protocols. Rather than relaying data via intensive routing procedures, the distributed technology can spread data source over multiple cooperative components to facilitate the data access. Particularly, decentralized ledger technology (DLT), which is in essence a distributed technology, incorporates all the participants to maintain and synchronize the full copy of data. Coupled with the consensus mechanism, it guarantees the preservation of trustworthy data. These two key features of DLT contribute a more reliable data delivery. However, in VANETs, due to the locomotion of vehicles, the participants of DLT frequently adjust their physical connection, and thus interrupt their data transmission. In this paper, we propose a novel framework, where the VANET is built upon the virtualization of DLT (vDLT), to achieve seamless and reliable data transmission. In the proposed framework, components in VANETs are equipped to run vDLT nodes, which disseminate data in the virtualization layer; thus, the variation of physical layout is transparent to the data transmission via vDLT. Simulation results are presented to show the effectiveness of the proposed framework.

4.2.1 Proposed Framework

In this section, we describe the design of the proposed framework of VANETs using vDLT to consolidate data delivery amongst vehicles in the ad-hoc domain. The interaction between ad-hoc domain and the Internet still works in the traditional fashion, which employs mobile devices and routing protocols.
4.2.1.1 Architecture

The architecture is constructed based on overlay networks, and is composed of the elements running the vDLT system in the virtualized environment. The overlay network is built upon physical topology, by virtualizing the network resource in hardware layer [30]. To clarify the architecture, transformation from physical network to overlay topology, connection fashion and roles of participants in the overlay network, are described in Fig. 4.9.

As depicted in Fig. 4.9, the transformation of network using virtualization creates more logical connections above physical topology to stabilize the inter-connection between participants. As shown left in Fig. 4.9, connection in the physical underlying network is created in the ad-hoc fashion. Data traffic travels on routing paths, and vehicles on the move require to repeatedly compute the best routes for efficient data transmission. On the contrary, the overlay network allows vehicles to maintain multiple logical connections. As a result, despite losing physical connection and missing available routes, data transmission can still survive by passing different stable
logical paths.

In the architecture, participants are embedded with vDLT nodes, which use virtual network devices for communication in the overlay network, and virtual storage and computing resources for running vDLT system. Various functions are provided by the platform of vDLT nodes, and a collection of them are enabled in a participant according to its role in the architecture. Generally, all participants can be grouped into three logical scopes of roles, i.e., Producer Domain, Seed Domain, and Consumer Domain.

1) **Producer Domain**: The centered scope, which is composed of RSUs, provides the production and verification of data. Working as producers of vDLT, the RSUs are able to manipulate data storage (aka. ledger), and with decentralization mechanism, all the RSUs, including producer candidates, preserve the same copy of ledger through data synchronization. RSUs are connected in a logical mesh P2P network, to increase the reliability of data synchronization. Since they can be accessed by their neighboring vehicles, any data over a wider range can be pushed to or pulled from the ledger by means of any RSU. In addition, RSUs can exchange data with vehicles. For the purpose of resource efficiency, vehicles are disallowed to engage in this domain.

2) **Seed Domain**: The neighboring vehicles of RSUs and those registered as Seed Vehicles (SVs), form this scope. These SVs synchronize data with RSUs and other counterpart vehicles, and involve validating and relaying data. Due to the locomotion, no specific network topology is used in this domain. To avoid cumbersome re-connection, vehicles firstly put an area of RSUs into their peer lists, and their sense of RSUs recorded triggers the logical inter-connection. Besides, vehicles send light-weight messages to prove their own authenticity for on-demand seed-to-seed or seed-to-RSU connection. Through logical links, the SVs staying in this domain can constantly distribute the access of data for
consumers. However, a SV disconnecting with other seed peers or RSUs will lose the role of provisioning and validating data, which can be withdrawn through Hypervisor.

3) **Consumer Domain**: This domain contains Seed Domain as its subset, because vehicles in this domain exchange data with SVs or RSUs, and both Consumer Vehicles (CVs) and SVs require this function. CVs, as without data synchronization, remotely employ SVs to interact with the ledger of vDLT. Contrarily, SVs can handle the interaction locally. Inside vehicles, their own client and wallet applications of vDLT are installed for security. Besides, the consumer-to-consumer connections are established by preset routing protocols, while consumer-to-seed data delivery require CVs to broadcast request and obtain response from all the neighboring SVs, so as to access the ledger via multiple links. Note that, CVs registered for data validation can become SVs once reaching RSUs or SVs, and similarly the roles can be enabled through Hypervisor.

The three above domains, which are designated in the overlay network, constitute the proposed architecture. Virtualization improves the scalability of system. For example, vehicles can agilely adjust their functions through Hypervisor once crossing the boundary between Seed Domain and Consumer Domain. The overlay network structure sustains the collaboration amongst SVs and RSUs, such that reliably enlarges the range of data provisioning. The logical connection can reduce routing procedures and influence of underlying physical network, and thus facilitates seamless data delivery. With DLT, multiple RSUs cooperatively manipulating data, the overhead of procedures, along with energy consumption in a single RSU, will be relieved. Meanwhile, with data decentralized over various vehicles, passengers anywhere can more rapidly supply or obtain the important information to help clarify the traffic condition. The mechanism of creating overlay network upon virtualized devices will be explained together with the design of components.
4.2.1.2 Components

A logical component is given to express a set of relevant functions, and the domain of participants determines their required components. The design of the components, and their relationships are illustrated in Fig. 4.10. According to Fig. 4.10, vDLT nodes running inside RSUs and SVs are installed in the virtualization layer, which incorporates logical network resources, such as routers, switches, and interfaces, and runs a system for data processing with virtual computing resource. The underlying hardware layer, which mainly involves network interfaces and sensors, supports the functionality in the virtualization layer. But most importantly, the resource management in these two layers is isolated. The following elaborates the components functioning in the architecture.

**Logical Router**, a component serving in RSUs, provides ports for logical switches to attach, as well as the peer ports for other routers. The peer ports in two routers
create a logical connection between two subnets. With more peer ports, a logical router can reach more subnets. A RSU running out of power, can transfer all the logical devices to its neighboring RSUs, such that the overlay network can preserve the same routing tables.

**Logical Switch** can be executed in RSUs and SVs. The switch inside RSUs attaches to the local logical router, while the one in a SV to the neighboring RSU. A logical switch open a number of ports to connect with virtual interfaces. When a vehicle leaves one RSU and approaches another, the physical link accordingly switches. By transferring the related router port from the previous RSU to the current one, the logical switch can keep the same configuration. Since the MAC address is consistent, traffic via this switch persists. If the vehicle cannot reach the producer domain, its switch will be withdrawn by the last connected RSU and assigned to another candidate. Besides, the switch in a dying RSU can float to other counterparts.

**Logical Interface** is enabled in RSUs and SVs, and is attached into a logical switch. The neighboring SVs of RSUs, connect their virtual interfaces to the local logical switch, whereas the others merely connect their interfaces to a remote switch. Once the interface loses connection with the remote switch, its port and the switch port will be discarded. On the other side, if a vehicle without logical switch, detects another one existing remotely, it will obtain a new logical interface to attach the new switch, along with a logical P2P link created in between. Eventually, all the outgoing traffic can also use the new interface, and the new provider can take over data provisioning. Meanwhile, this shift of switches can inform the previous switch provider of adding a new P2P link with the new interface, and hence the traffic keeps consistent.

**Data** is synchronized in blocks by RSUs and SVs. Based on the block header, blocks are logically chained up to form a so-called BC [137]. Fig. 4.11 illustrates the structure of BC. In Fig. 4.11, block data is composed of traffic data, such as a picture and video clip. Each piece of traffic data is indexed with the area code and time
Figure 4.11: A general structure of BC used in VANETs.

stamp, and is related to a hash value. All the hashes of traffic data will be calculated to be the hash of block data. In addition, traffic data is exchanged between vehicles and RSUs via SCs, which can define the action on and parameters of related traffic circumstance.

Consensus employs DPoS mechanism [86], which uses BFT algorithm to determine the trust of data and uses random voting to eliminate centralized control. The votes are contributed by SVs and RSUs to assign a RSU to produce blocks. In process, SVs tend to vote their closest RSUs, while RSUs will vote themselves. This scheme can improve the efficiency of block production. Furthermore, all the RSUs participate in consensus, and store the data which obtains the agreement amongst most of RSUs. For example, in the case that the ledger in some RSUs is distorted, the hash of upcoming traffic data, which is generated in these victim RSUs will be inconsistent with other peers. It means that, the data from this RSUs is untrustworthy. Without the collaborative consensus, it is risky to count on the producer of data to determine the trust of data.
<table>
<thead>
<tr>
<th>Action</th>
<th>Parameters</th>
<th>Permission</th>
<th>Sign</th>
<th>Indexing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSH</td>
<td>AREA</td>
<td>WRITE</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>TIME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LICENSE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PULL</td>
<td>AREA</td>
<td>READ</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td></td>
<td>LICENSE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Configuration of SCs for procedural data exchanging.

**API and Validation** are the functions provided by SVs and RSUs to jointly secure the data. API is used for exchanging data with CVs, while validation for filtering out the dishonest data. Traffic data travels in the form of a series of dedicated SCs, which is shown in Table 4.3. The SC set supports the actions of PUSH and PULL of traffic information along with the main parameters of AREA, TIME and LICENSE. According to Table 4.3, in the SCs to be developed, LICENSE of a vehicle is taken as the account of transactions, AREA as primary key and the combination of AREA and TIME as index. API component receives traffic data from a specific port. The validation component verifies the signature of data belonging to LICENSE, and checks the conflict of index in the ledger during block production. If no conflict, the block will be sent to the connected RSUs by vehicles or the local system by RSUs. The block subsequently will be put into real chain by the on-duty producer and eventually into consensus stage.

**Client** is a sub-system of vDLT node. The client system incorporates the functions calling the API in SVs and the wallet application. To reduce communication cost, wallet is locally managed by the CVs instead of a remotely shared component. To exchange data of traffic with vDLT, CVs should open accounts in advance, and follow the format of SCs to construct transactions. For each submission of transaction, CVs
will receive a receipt indicating the result.

Generally, virtualization and DLT jointly can improve the reliability of data transmission in VANETs. The scalability of virtual resource facilitates the relevant management and mitigates the issues caused by physical variation of VANETs. Logical network components can be flexibly spawned, scheduled and modified, and thus jointly establish the overlay network for VANETs. With their information shared between all the RSUs, any variation of network condition triggers RSUs to notice and conduct the related nodes to reschedule the logical resource. Components of consensus and data work in a virtualized system, a so-called guest system, which is built and managed via Hypervisor. While the domain of vehicles changes on the move, the functionality of vehicles need to be scaled. This adjustment is made to the guest system, without influencing the host. Therefore, with components running in the virtualization layer, despite the variable structure of VANETs, a stable architecture can be constructed, which greatly improves the reliability of system.

4.2.2 Simulation Results and Discussions

Instead of implementing in real vehicles and roadsides, the framework is simulated based on computers that are connected in a Wi-Fi network. To simulate the locomotion of vehicles, firewall is used to automatically change the inter-connection according to time series, which is calculated randomly. Therefore the physical topology is refreshed from time to time, and accordingly the allocation of virtual resource changes. In this environment, the simulation of proposal utilizes vDLT as the base, and develops the logics designed above to improve the data delivery.

4.2.2.1 Environment Setup

To prepare the simulation environment, there are a series of physical or virtual devices which need to be enabled in advance. In this simulation, a number of computers
equipped with Wi-Fi devices and installed with relevant software tools, are used to serve as RSUs, SVs and CVs. Their configuration is shown in Table 4.4.

<table>
<thead>
<tr>
<th>Node</th>
<th>Number</th>
<th>Specs per Node</th>
<th>Producers</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSU</td>
<td>4</td>
<td>CPU: 4core, 2.66GHz&lt;br&gt;MEM: 32GB&lt;br&gt;B/W: 100Mb/s</td>
<td>24 (6 per RSU)</td>
</tr>
<tr>
<td>SV</td>
<td>6</td>
<td>CPU: 2core, 2.66GHz&lt;br&gt;MEM: 8GB&lt;br&gt;B/W: 100Mb/s</td>
<td>N/A</td>
</tr>
<tr>
<td>CV</td>
<td>2</td>
<td>CPU: 2core, 2.66GHz&lt;br&gt;MEM: 8GB&lt;br&gt;B/W: 100Mb/s</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 4.4: Configuration of the platform for experiment.

Since vDLT is a framework that can embrace various BCs and virtualization platforms, the environment of simulation is built up in three major steps to make vDLT functions.

1) Virtualization of system is implemented based on Docker [121]. Docker is a technology of containerization, which provides a light-weight virtualized environment for running application and API for management.

2) EOSIO [28], as a BC platform supporting DPoS, is installed in the container of Docker to being managed by vDLT system. To control the vDLT system, a series of scripts using Docker API will be created to interact with Docker daemon. In vDLT, each RSU host can run multiple EOSIO systems, so the 21 producers will be created.

3) Overlay network is accomplished by setting up a Open Virtual Network (OVN)
through OpenvSwitch [62] [88]. As the OVN supports SDN, the proposal will
distribute its northbound and southbound databases over RSUs, and update the
database through scripts, which are developed based on the commands provided
by OpenvSwitch.

4.2.2.2 Topology Variation

In terms of simulating the change of physical topology, a vector is used to indicate
the connection and disconnection between a participant with the others. The vector
is shown in Eq. (4.5).

\[
\text{Link of Node}_i = \left[ l_{i1} \ l_{i2} \ \ldots \ l_{ij} \ \ldots \ l_{in} \right], \quad (l_{ij} \in [0, 1]) \tag{4.5}
\]

For Node\(_i\), \(l_{ij} = 1\) indicates that it connects with Node\(_j\), while \(l_{ij} = 0\) means
disconnection. At a time point, a matrix will be constructed in terms of connection or
disconnection between all Participants. Matrix (4.6) gives an example to illustrate a
topology comprised of 6 participants.

Based on the matrix like the above example, a vDLT script to change the firewall
rules will be generated and executed in all the participant hosts. Once the firewall rules
are enabled, the status of physical network interface will change. Correspondingly,
the OVN script in vDLT, which continuously listens to the communication interface,
searches network configuration in a global database of vDLT. This preset information
and the matrix will be jointly used to change the network topology, and finally to
update the database.

\[
\begin{bmatrix}
RSU1 & RSU2 & SV1 & SV2 & CV1 & CV2 \\
RSU1 & 0 & 1 & 1 & 0 & 0 & 1 \\
RSU2 & 1 & 0 & 0 & 1 & 0 & 0 \\
SV1 & 1 & 0 & 0 & 0 & 0 & 0 \\
SV2 & 0 & 1 & 0 & 0 & 1 & 1 \\
CV1 & 1 & 0 & 0 & 1 & 0 & 0 \\
CV2 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

4.2.2.3 Evaluation

In evaluation, performance metrics to be used include throughput, delay and packet loss. The baseline is created by implementing AODV [19] and DSDV [39] routing protocols. AODV periodically broadcasts HELLO packets to build the new routing paths; while DSDV broadcasts its routing table. The same time series and matrices of changing physical topology are used in the baseline and the proposal. The evaluation result will be explained together with figures below, which respectively compare the delay of packet, throughput and average packet loss ratio.

Fig. 4.12 indicates the comparison of packet delay. According to Fig. 4.12, the proposal framework obviously performs better than AODV and DSDV in terms of the delay of packet, which is evaluated based on the request sent and response received in vehicles. Since there are various SV and RSUs providing access to data, any vehicles can rapidly interact with the ledger. Note that, the 500 milliseconds taken by EOSIO to produce a block and the time to verify the block have no effect on this delay, since this amount of time pertains to consensus, but not the response to requesters. Whenever or wherever receiving a request, vDLT promptly returns a
response to indicate the status of data. Actually, Transaction Per Second (TPS) is the performance metric to evaluate the efficiency of block production. In our proposal, the virtue of consensus is embraced in the framework to improve the trust of data, and thus to provide reliability for data delivery. Without routing procedures, the proposal indeed reduces the time spent on data transmission; thereby, vehicles can quickly communicate with each other. However, AODV and DSDV require to periodically broadcast HELLO messages or routing tables to update routes. Before the routing table turns available, outgoing traffic is queued or even discard, and hence the waiting time is longer. In addition, DSDV sends and updates the whole routing table at intervals; due to that heavier workload on larger data volume, oscillation is witnessed in its graph, and more time is spent on preparing routing protocols than AODV.

The result of throughput is plot in Fig. 4.13. Unsurprisingly, as shown in Fig. 4.13, the proposal mostly contributes the highest throughput due to the lowest delay. According the trend in Fig. 4.13, throughput of the proposal keeps increasing until the middle time point, for the reason that, the amount of blocks to be synchronized
is small and consumes less bandwidth. But in the latter half period, more blocks are exchanged in the network, and hence the throughput decreases. With data synchronization finished, more network resource becomes available, and hence support faster data transmission. Because of the virtual network, the data transmission can proceed constantly at most of time, except when all the vehicles are individually isolated. In routing protocols, searching and rebuilding routes are necessary; but in the proposal, routing scheme is replaced by configuring logical network devices. By simply transferring switches, ports of switches and routers, or creating logical interface through a piece of commands, the ongoing data transmission keeps moving on without much interruption. Therefore, better throughput is obtained. Similarly, throughput in AODV and DSDV, as affected by delay, is witnessed lower than the proposal one. In addition, AODV in this evaluation outperforms DSDV. Actually, in the experiment, just 12 physical hosts are used, the number of messages exchanged in AODV is considerably small, and this traffic volume is smaller than the one in DSDV,
Along with less bandwidth occupied.

After the simulation, the numbers of packet sent and received are computed to evaluate the rate of packet loss. This comparison is illustrated in Fig. 4.14. In Fig. 4.14, the column related to the proposal is shorter than the others. First of all, the figure of the column is really small, but still cannot be ignored. Despite floating virtual resources can most possibly stabilize the connection, physical network resource are in essence the real one supporting the data transmission, and its variation still have an influence on the upper applications. Nonetheless, the use of logical network devices in our framework avoids frequently rebuilding the physical configuration inside participants, such that preserves the underlying reliability and consistency, and meanwhile reduces overhead for transmitting data. This steady and comfortable environment greatly mitigates the packet loss, where data transmission dodges interruption caused by missing or computing routing information, but instead proceeds smoothly along with prompt and simplex reallocation of logical resources.
In comparison, it can be seen that, the average packet loss ratios of the other two approaches, are much higher, even though just 12 physical machines are being used. In AODV and DSDV, physical resources are directly used for transmitting data, and they are also shared by the routing scheme. The change of physical topology and the update of routing tables stall the in-progress data traffic, and the limited re-transmission cannot always protect the packet delivery. More seriously, as the number of nodes increases, the packet loss ratio surges correspondingly. But as opposed, we can see the potential of our framework. Since with more nodes taking part in the proposal system, the more data accesses and providers can be given to escort the data delivery, eventually, packet loss will be controlled to a considerably low level.

4.2.3 Remarks

In VANETs, routing protocols consume a lot of network resources to frequently establish routing paths, since vehicles move continuously. The adjustment of connection according to the new routing table causes choppy data transmission. Therefore, we presented a novel framework in this paper, which aims to provide a stable topology and a wider range of data provisioning so as to consolidate the data transmission. The framework is based on the integration of vDLT and VANETs. In the proposed framework, network resources, such as router, switch and interface, are all created by abstracting hardware resource; and further provide flexibility and scalability to simulate or to implement various DLTs. Specifically, we installed Docker, EOSIO and OpenvSwitch to simulate a VANET using vDLT for data sharing. This integration can eliminate the influence caused by variable structure of VANETs, and hence consolidates the data delivery inside. To mimic the locomotion of vehicles, we designed a matrix to indicate the state of inter-connection between participants. Following this matrix, vDLT sets Firewall rules accordingly to make the topology change. This change subsequently triggers vDLT to execute OVN scripts, which are used to manage
the virtual network and its resources. The matrix of physical topology and the configuration of logical network, are saved in the global database of vDLT. Simulations results were presented to show the effectiveness of the proposed framework. From the results of simulations, we observed that both the delay of packets and packet loss ratio reduce, and higher throughput is generated. Therefore, the framework is feasible to consolidate data delivery in VANETs.
Chapter 5: Sharing Visual Traffic Information via vDLT

The extensive use of vehicles nowadays, especially with the emergency of autonomous driving, urges the improvement of traffic safety. Prevalent approaches, such as Global Positioning System (GPS), Internet of Things (IoT) system and even Artificial Intelligence, have demonstrated their strength in preventing road accidents, and their remarkable achievement is inseparable from the trustworthy and secured data acquisition. However, in vehicular ad hoc networks (VANETs), data transmission and storage is unreliable due to various constraints such as limited physical resource and unsteady topology. For that issue, distributed schemes are widely applied in VANETs to enforce multifold protection on vehicular data, and in particular BC becomes preferred as it implements the real-sense distribution using consensus algorithm and distributed ledger. To this end, we propose a system in this paper, which employs BC technology to secure the sharing of traffic information and hence holds profound significance for intelligent applications. Our system focuses on the real-time visual traffic information which is uploaded into BC in frames via SCs. The sequence of frames is considered for integrity verification prior to the consensus and persistence in BC. Besides, for the efficiency and reliability of uploading, the system dynamically adjust the transaction volume in terms of the frame types. Transcoding frames is supported in BC, with the target format defined via SCs. The fault tolerance and immutability of BC solidly protect the traffic information against vandalization, and hence vehicles can confidently use the information for safety guidance.
5.1 System Design

This section will explain the architecture and algorithms of our system. In our design, visual traffic information is the basic data object which mainly involves videos and images are captured by the cameras in VANET nodes. This multimedia data is unicast via the RTP protocol inside VANETs, which runs over UDP but enables packet loss and reordering management [100]. Besides, the design applies DPoS consensus algorithm [86], which is able to reach more than thousands of transactions per second (TPS). Therefore, with DPoS and RTP, the design is able to achieve both reliability and efficiency improvement.

5.1.1 Architecture

The architecture embraces a BC network for reliable sharing of traffic information in VANETs, which is shown in Fig. 5.1. According to Fig. 5.1, the major part of BC inside the VANET is formed by vehicles, traffic lights and roadsides which commonly expose functions to serve as the access point of BC, and particularly roadsides and traffic lights can also be the block producer and validator along with block synchronization. In addition, since VANETs are connected to the Internet through base stations (BSs) and the relevant gateways, the BC is extended to the outer of VANETs, where many servers are placed as participants to empower the capability of BC.

Client nodes, such as vehicles, traffic lights and roadsides are equipped with cameras to capture the traffic circumstances and with displays to clarify the traffic condition of intended location. For example, vehicles traveling around an accident scene can record a video, traffic lights are triggered to take the picture of vehicles running a red light, roadsides are able to monitor the speeding or impaired driving. These traffic events imply the malicious traffic condition, and thus by querying the traffic information in advance, drivers will exercise caution to avoid accidents. For example, a driver is
Figure 5.1: The proposed system architecture composed of a VANET and a BC Network

going to a sharp turn, but there is an off-sight accident behind the corner due to being blocked by woods, and thus the driver is possible to crash into the accident scene. But querying the trustworthy traffic information in advance can help the driver prudently avoid the potential risk. Therefore, the VANET nodes are encouraged to report the traffic information, which is preserved by using BC for reliability and trust protection.

In order to exchange traffic information between client nodes and BC system, a range of traffic lights and roadsides are established as the access points to BC. As the BC participants, BC access points carry wallets of account information, and support the access to BC storage, the transmission via RTP, as well as the conversion between
traffic events and transactions for clients. In our system, traffic lights and roadsides are mainly considered as the BC access points, since they have fixed geographic locations and steadier resource suppliers than vehicles. By means of stable BC access points to collaboratively exchange data, the moving vehicles can achieve reliable and efficient data exchange with BC.

To be uploaded into BC, traffic information is unicasted via RTP from client nodes to BC access points for transaction generation. The conversion and uploading of transactions count on SCs which is related to BC accounts. Besides, BC access points use their account information to generate signature to protect the transactions. Vehicles without BC accounts are unable to handle transactions, and thus they require BC access points to achieve the indirect interaction with BC; while the nodes, such as traffic lights and roadsides provide BC access points, and hence they can use the BC accounts and call the local functions with the input of traffic events captured by their cameras to build and upload transactions. In BC, the transactions undergo distributed validation, consensus and storage, and eventually achieve the real-sense protection.

Inversely, traffic information is downloaded via BC access points with a SCs used to initiate the downloading request, and transcoding traffic information is implemented in clients. Since without BC accounts, vehicles are unable to obtain SCs, and hence depend on BC access points for data downloading from BC. The traffic events downloaded is sent from BC access points to vehicles via RTP unicast. On the other side, nodes running both clients and BC access points can construct the downloading request transaction, therefore they just use the local functions to exchange data with BC, and playback the traffic events in no need of RTP streaming. The trustworthy traffic information from BC can help clarify the traffic condition, such that can help improve the traffic safety.

For efficiency improvement, except being BC access points, traffic lights and roadsides can be designated as the nodes of BC producers, validators and feeds.
These kinds of participants commonly carry the whole copy of BC data via data synchronization with each other, and thus can accelerate the data sharing inside VANETs. BC producers are responsible for the creation and consensus of blocks, validators focus on the validation of block, and feeds provide clients the read-only access to storage. With the whole copy of BC data held by VANET nodes, BC access points can accelerate the data retrieval from BC system, and hence the query of traffic information. Furthermore, with VANET nodes voted as producers or set as validators, the uploading of traffic information to BC and its final preservation can be accomplished faster.

As the more participants are involved, the more capability and security can be achieved by BC system. Hence, in our system, a number of servers are placed outside the VANET, to contribute their effort on block production, consensus and preservation. Through the BSs, the VANET is connected with the network of servers. Due to side effects caused by fault tolerance, such as re-connection and re-computation after state changes of BC nodes [27] [118], using automobiles as BC participants is inefficient. As considering traffic lights and roadsides have stable resource supplier in VANETs, their connection with the external servers is feasible to extend the BC network and enhance the BC security.

5.1.2 Model Design

Our system is designed based on a 3-Tires model, which is shown in Fig. 5.2. According to Fig. 5.2, the top layer encompasses the client application (CLI) and its multimedia traffic information (TINFO) which is created by cameras and consumed by displays, the middle layer pertains to the services of BC access points (BCAPs) which handle the transaction of traffic events (TTRXs), and the bottom layer illustrates the block storage in which the services of BC producers (BCPDs) create blocks and synchronize with other BCPDs, and logical components of BC feeders (BCFDs) and BC validators.
(BCVDs) for distributed consensus and storage. In general, by going from the top layer to the bottom, traffic information is eventually preserved and distributed over the BC network, while in the inverse direction is restored from BC.

They first layer illustrates the flow of TINFOs TINFOs are originated from cameras and finally reaches the display terminal. The structure of TINFOs includes codec and RTP contexts, sender ID, time and GPS information, and the compressed content of multimedia traffic event with the watermark of multimedia metadata. The transfer of TINFOs across networking nodes in VANETs requires RTP unicast. The integrity and consistency of multimedia is verified based on the RTP timestamp increment between frames. Eq. (5.1) is used to calculate the value of RTP timestamp increment \[\Delta TS = \frac{Freq_{sampling}}{FPS}\]  

\[\text{(5.1)}\]
In (5.1), $\Delta T_S$ represents the incremental value of RTP timestamp, $Freq_{sampling}$ is video sampling frequency and $FPS$ is frames per second to be sent. Since $Freq_{sampling}$ and $FPS$ are constants during multimedia streaming, $\Delta T_S$ is unchanged and thus can be used to check the consistency of frames. Additionally, cameras and displays in this layer are controlled through multimedia tools and APIs, such as FFmpeg [8] and VLC [123], to support the capture, transcoding and playback functions. Clients without BC accounts are placed in this layer to directly use multimedia tools and APIs to manage TINFOs.

The second layer targets the TTRXs and BCAPs In this layer, BCAPs bridge the CLIs and BC system to support uploading and downloading of TINFOs within TTRXs. The relationship between CLIs and BCAPs is saved in BC for tracing the activities of CLIs and to support the trust management of CLIs. On uploading, TTRXs are created to wrap up a batch of TINFOs which are received from CLIs according to a dedicated SCs description. TTRXs are protected with a signature which is generated jointly by BC account key and the TTRX data. If a TTRX is vandalized, the signature will become unmatched, and thus the trust of internal TINFOs is protected. On the other side, via BCAPs, the records of TTRXs can be downloaded from BC, subsequently parsed to restore the TINFOs, and finally received and displayed by CLIs. Besides, the RTP timestamp increment calculated from the RTP context, is used along with RTP sequence by BCAPs to verify the consistency of remote frames. The discontinued or disordered frames imply the dishonest traffic information which might be vandalized, and BCAPs will report the issue and related CLIs to BC for further trust management. The followings list the services deployed in BCAPs.

- The first service supports the discovery of available BCAPs for remote CLIs via multicast. On receiving a discovery request in a multicast group, BCAPs examine the system resource, generate and return a related response to CLIs.
The pair of request and response is cached by BCAPs for later data exchange.

- Another two services are exposed to handle the uploading of TINFOs, with one as the RTP receiver, and the other as TTRXs converter. Both services serve CLIs in the one-to-one way, and the RTP receiver serves only the remote CLIs while the TTRXs converter handle all the TINFOs along with frame verification.

- The last two services aims at the downloading procedures. As similar to the uploading process, these services and CLIs are mapped uniquely, and one of them communicates with remote CLIs via RTP. As opposed, the BCAPs provide TINFOs restoration and send the streams via RTP.

In addition, BCAPs support security features in communication using the mature techniques, such as SSLv3 [29] and TLS 1.2 [95]. Generally, the group of BCAPs collaboratively and securely relay TTRXs between CLIs and the BC distributed storage.

The third layer involves the block arrangement in BC storage and the participants of BCPDs, BCFDs and BCVDs  
Synchronization of blocks is periodically conducted among BCPDs, BCVDs and BCFDs to ensure the integrity and consistency of their BC storage, such that, parts of BCFD nodes in the VANET can also serve as BCPDs or BCVDs to handle authenticity of BC data. The DPoS consensus in our system refers to the BFT-DPoS algorithm [124], in which fixed time intervals are arranged for block production, and a new block will be sent out for validation and consensus before the next block production. With the help of validators and BFT algorithm, since without the next block production to confirm the current block, the time on block confirmation is significantly reduced. Eqs. (5.2) and (5.3)
explain the time spent on the preservation of frames inside BC.

\[ T_{\text{con}} = T_{\text{bint}} + \max(T_{\text{brcv}} + T_{\text{bchk}} + T_{\text{bstb}}) \]  
(5.2)

\[ T_{\text{delay}} = T_{\text{con}} \left( \frac{T_{\text{fgen}} + T_{\text{frcv}}}{T_{\text{bint}}} + 1 \right) - T_{\text{fgen}} \]  
(5.3)

In (5.2), \( T_{\text{con}} \) is the total time to confirm a block, including \( T_{\text{bint}} \) as the time spent on producing each block, \( T_{\text{brcv}} \) on receiving a block in BC nodes, \( T_{\text{bchk}} \) on validating the block, and \( T_{\text{bstb}} \) on exchanging the view state of block. Eq. (5.3) explains the delay to save a frame BC, in which \( T_{\text{fgen}} \) represents the time point of a frame generation, and \( T_{\text{frcv}} \) is the time spent on delivering a frame to the BC producer. According to Eqs. (5.2) and (5.3), \( T_{\text{fgen}} \) is determined by the sampling rate of cameras which is usually constant, \( T_{\text{bint}} \) is another constant value, and \( T_{\text{bchk}} \) depends on computation resource, therefore, reducing delay focuses on the time in transmission. Due to the transmission latency, there remain some frames which are created in a production interval, unable to reach the current block, such that they have to be pushed into the next block which will downgrade the performance of data sharing. In our system, VANET nodes participate the validation and consensus of blocks, and the BC participants are connected in a mesh topology to provide multiple shortest routes, therefore it is effectual to accelerate the data transmission over BC.

### 5.2 System Development

According to the proposal design, there are several services and SCs to be developed, and our development applies modularization and multi-threading concurrency to boost the scalability and efficiency of system. Among the components involved, those in the middle layer gain the most attention of development. BCAPs need to provide various services to bridge the applications in CLIs and BC system, and correspondingly, a set of SCs together with permission assignment are developed to define the data conversion
Figure 5.3: The services for collaboratively discovering BC access points between TINFOs and TTRXs for BCAPs. In general, our development uses vDLT as the BC platform, and focuses on the FFmpeg APIs [85] to support multimedia streaming and processing along with multimedia protection by watermarking [18].

### 5.2.1 Discovering BCAPs

Clients without BC accounts require a discovery service to retrieve a list of remote BCAPs for the succeeding operation on TINFOs and TTRXs. Fig. 5.3 illustrates the components developed to support the discovery of BCAPs. In Fig. 5.3, CLIs and BCAPs are logically isolated, and communicate with each other via networking or local system. For remote communication, due to the mobility of vehicles, TCP protocol consumes extensive resource to manage the transmission and connection, therefore, UDP sockets are preferable for this short-term and one-time data exchange. Furthermore, multicast [132] is used as it facilitates CLIs to quickly find the available...
BCAPs which are seated in the same multicast group.

To initiate discovery request, CLIs need to check the existence of local wallet. If without wallets, CLIs have to fill in the requests of REQ-PUT-TINFO and REQ-GET-TINFO to respectively proceed with the downloading and uploading of TINFOs. The REQ-PUT-TINFO structure mainly includes the parameters for capturing and streaming videos, such as codec, sampling frequency and bit rate, while REQ-GET-TINFO provides the RTP address and port for receiving videos which are the available resource in system. The initial CLIs need to provide ID and time information inside request messages to identify the connection establishment for data exchange. Eventually, remote CLIs use sockets to send out requests, and to wait for responses with timeout control, and cache the relationship and content of requests and responses for redundancy check and connection establishment. For CLIs with wallets, they are installed with BCAPs in the same node, and hence they can locally load up the functions of BCAPs with the inputs, such as ID and time information.

BCAPs receive remote requests through their sockets. With the request parsed, BCAPs generate the response of ANS-PUT-TINFO or ANS-GET-TINFO which is mapped with the request of REQ-PUT-TINFO or REQ-GET-TINFO. To reply ANS-PUT-TINFO, BCAPs search the available RTP address and port in system; if failed, they will forward the request to other nodes in case the TTLs in multicast are insufficient, otherwise return the ANS-PUT-TINFO with RESULT and a random UUID to senders which is uniquely assigned to identify a sender. The ANS-GET-TINFO is used to indicate the multimedia parameters of RTP streaming for downloading TINFOs in CLIs. ID and time information are also required to return by BCAPs, in order to help CLIs with connection establishment and redundancy check. Besides, the security features are enabled in BCAPs by caching the request and response.

In general, as seen from Fig. 5.3, both CLIs and BCAPs collaboratively relay the requests and responses to extend the multicast range for the original CLI reaches sufficient available BCAPs, and meanwhile use timeout to manage the resource
efficiency. Owing to the mesh topology used, CLIs and BCAPs possibly receive repeated requests, therefore the caches in both sides can be used to skip the redundant message and hence to further improve the resource efficiency.

5.2.2 Uploading TINFOs and TTRXs

CLIs employ BCAPs to upload traffic events to BC, and the components and procedures developed are shown in Fig. 5.4. The information cached in the discovery process is used by CLIs and BCAPs to create the connection for uploading. There are two threads and a message queue used by both CLIs and BCAPs to improved efficiency and reliability of processing TINFOs and TTRXs. In CLIs, the camera thread calls FFmpeg APIs to control the camera to capture traffic events, the sender thread convert traffic events into TINFOs and send them to BCAPs, and the message queue is set in between for concurrency control on deal with the traffic events. While in BCAPs,
one thread named receiver receives TINFOs via RTP and write them to the BCAP message queue, while the other thread, uploader, reads TINFOs from the queue and convert them into TTRXs followed by being uploaded into BC via a P2P interface. In addition, UDP communication and RTP streaming are jointly used for efficiency improvement.

Initially, CLIs without local wallets send a request including the UUID and time information to notify BCAPs to check and start the two threads via UDP sockets, while CLIs having accounts locally boot up the uploader thread in BCAPs for TTRXs generation and uploading, and requester CLIs wait for the result from BCAPs. Subsequently, CLIs verify the result, and exit if seeing errors, otherwise proceed to create their own two threads to generate the TINFOs of traffic events. The successful connection between CLIs and BACPs is cached in both sides, and eventually uploaded via TTRXs to BC, which means the CLIs register in BACPs to interact with BC. The continuous frames are buffered into a message queue, and meanwhile are read by the sender from queue. The sender thread firstly extracts the RTP and codec contexts, and build their TINFOs followed by their transfer to BCAPs. By repeatedly reading the queue, the sender thread retrieves the frames in which it embeds the GPS and time information by watermarking, then build and finally transfer the TINFOs. For non-wallet CLIs, the transfer of TINFOs counts on RTP unicast and reaches the receiver thread in BCAPs, whereas by CLIs with accounts, TINFOs are directly written into the message queue in BCAPs.

As the counterpart, BCAPs create UDP sockets to accept the remote request. On receiving a remote request, BCAPs extract the UUID, and search the address and port in the cache, which are previously arranged to a sender with the UUID. With the address and port, BCAPs attempt to start threads and return their status. While being called by local CLIs, the information of ID and time is input as parameters, and the UUID is also cached to avoid its repeated generation. In BCAPs, the threads are assigned to the CLIs according to UUIDs and the assignment is cached for security.
and connection check. The receiver thread calls FFmpeg APIs to receive TINFOs from RTP unicast which are then written into the message queue. Simultaneously, the uploader thread listens to the queue, read the incoming TINFOs and extracts the contexts. After that, it verifies the consistency of frames which comes from RTP layer by using their RTP contexts including timestamp and sequence, and the result will be written into TTRXs. The CLIs seated in the same node as BCAPs are always trusted without frame verification, since they avoid the attacks in networks. The contexts and sender UUID are subsequently converted into TTRXs by using the account key and SCs of BCAPs, followed by the uploading of TTRXs to BC. With the related SCs retrieved and TINFOs read out continuously, the uploader thread keep constructing frame TTRXs and their uploading to BC. Eventually, TTRXs are protected with signatures, and transferred to BCPDs via local P2P interface.

Besides, in our system, the multimedia compression incorporates the uses of I-frame, B-frame and P-frame. To scale the transaction volume, the attribute of frame length is the critical point. Since I-frame contributes the largest data size, the transaction holding I-frame remain the least resource, especially the storage space. Thus, according to the batch size defined in advance and the frame type, together with compressed frames in transmission, our system adjusts the frame number to prevent the loss and the delay of traffic information, and hence to improve the usage rate of transactions and meanwhile the robustness and efficiency of system.

5.2.3 Downloading TINFOs and TTRXs

As similar to uploading, CLIs to download TINFOs require the BCAPs, but with the opposite data flow, and Fig. 5.4 shows the development of downloading services. Multi-threading and message queuing are also used in this process, which two threads and a message queue in both sides of CLIs and BCAPs. CLIs use their two threads to respectively receive and display TINFOs from BCAPs, while BCAPs download
The services for downloading traffic information via BC access points

TTRXs from BC and perform their conversion and transfer of TINFOs for CLIs. The message queue in both sides is used for buffering data and synchronizing the access. Besides, P2P interface is used for data sharing among BC participants, and UDP and RTP for remote communication between CLIs and BCAPs.

Also starting with request sent to BCAPs and then getting the running state of their services, CLIs boot up the display thread and receiver thread. The CLI-and-BCAP relationship in this request is also uploaded to BC. The existence of wallet also impacts the utilization of BCAPs services, which means that non-wallet CLIs require BCAPs to boot up corresponding sending and downloading services, whereas wallet CLIs just need the downloading support. More precisely, non-wallet CLIs receive TINFOs via the RTP protocol, wallet CLIs retrieve TINFOs from the queue in BCAPs. With the TINFOs written into the message queue in CLIs, the display thread can obtain the TINFOs and perform transcoding to shown the original traffic information. In addition, the RTP context in CLIs is only available when frames are received via

88
Figure 5.6: The set of collaborative SCs for traffic event sharing.

RTP transmission.

In BCAPs, there are downloading thread and sending thread which respectively get the records of TTRXs from BC and restore and send their TINFOs to CLIs. Before getting TTRX records from BC, BCAPs need to construct a new request TTRX, which includes the record indexes as query condition to accelerate the data retrieval from BC. Due to the P2P connection with BC, BCAPs can continuously and reliably receive the data immediately after the request TTRX is delivered to BCPDs, BCVDs or BCFDs. The received TTRX record are parsed into TINFOs which are then written with watermarking of trust information into the queue by the downloader thread. On the other side, the sender thread reads the TINFOs from the queue, and uses the codec contexts inside TINFOs for the decoding in CLIs, and the RTP contexts for tracing the history of traffic events. without transcoding.

In general, downloading procedures work in the similar pattern to uploading ones but with opposite data flow and without the verification of receiving frames. Remote
CLIs as traffic information reporter are probably act as attackers, and hence their frames require strict verification in BCAPs. But for CLIs querying traffic information, since the information is downloaded from BC, the trust is ensured; therefore for resource saving it is reasonable for CLIs to escape the frame verification.

### 5.2.4 Smart Contracts

A set of SCs cooperate to support the exchange of traffic information with BC. Fig. 5.6 illustrates the function, storage and the relationship of SCs. According to Fig. 5.6, three SCs are developed which are related to uploading, downloading procedures, and hold their own storage for TTRXs preservation. Besides, our system enables inter-call among the request, upload and download SCs, in order to achieve modularization and improve the scalability of development.

Request SCs acts as the initiator for CLIs to access BC via BCAPs. In this contract, two actions including INSERT and UPDATE are taken to add and modify requests. While BCAPs succeed in starting services for CLIs, they will use INSERT action together with inputs, such as CLI ID, BCAP ID, the hash of their concatenation (CLI-BCAP), option type and time, to upload the request to BC. Inside the request SCs, INSERT action writes the inputs into the request table which use CLI-BCAP as the primary key. The subsequent uploading and downloading requires the existence of CLI-BCAP, which is used to locate the uploading and downloading records related to the initial request.

For uploading process, upload SCs are needed to transfer frame content and frame contexts to BC. Context is extracted into a table of context while frame content into frame table. The two tables are connected with CTX-ID which is hash value generated with the context structure. CLI-BCAP is mandatory for the frame table to connect with request table. With INSERT action, the frames are push into frame table, and meanwhile locally call the UPDATE action in request SCs to update the total number.
of frames. Therefore, with the cooperation between request and upload SCs, the uploading activities and traffic information is clearly recorded. In addition, upload SCs provides SELECT action which can be called to query the frame and context tables.

The last contract is download SCs, which BCAPs use to initiate the downloading process in BC. The download table in this contract records the condition of query which pertains to time and location ranges, rather than the content of traffic information. The action of download SCs focuses on adding the query condition into the internal table, and calling the UPDATE action in request SCs to update the total frames of downloading. In terms of the storage design, download table refers to request table via REQ-ID and to frame table via time and location information. Therefore, download SCs requires the cooperation of request and upload SCs.

In general, the SCs are developed as light-weight services which benefit the agile development of system. Their inter-communication is performed using local system resource, and thus the efficiency is guaranteed, and moreover the resource optimization is convenient. The modularization used for SCs development can help with the scalable extension on system functionality without complicated design and modification.

5.3 Experiment and Evaluation

The experiment is conducted to evaluate the performance of system, including the effects of frame verification and watermarking, and the efficiency of handling traffic event video via BC. VANET nodes are virtualized in physical computers, and communicates in the application-level AODV routing approach. Traffic events are captured by IP cameras which support RTP streaming. Furthermore, vDLT is the BC platform running DPoS and BFT algorithms, the proposal services and SCs based on virtualized resource.
5.3.1 Environment Setup

The Environment is constructed by using a number of networking computers, in which IP Cameras are equipped, and the Docker suites [121] are installed to provide virtualized resource for running the vDLT system and simulated VANETs. The configuration of components is shown in Table 5.1.

<table>
<thead>
<tr>
<th>Node</th>
<th>Number</th>
<th>Specs per Node</th>
<th>Node Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>8</td>
<td>CPU: 8core, 2.66GHz</td>
<td>Physical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEM: 64GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B/W: 100Mb/s</td>
<td></td>
</tr>
<tr>
<td>Cameras</td>
<td>8</td>
<td>Max Resolution: 1080P</td>
<td>Physical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USB: 3.0</td>
<td>(in each)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FPS: 30</td>
<td>Computer)</td>
</tr>
<tr>
<td>Road sides</td>
<td>4</td>
<td>CPU: 1core, 2.66GHz</td>
<td>Virtualized</td>
</tr>
<tr>
<td>Traffic lights</td>
<td>4</td>
<td>MEM: 8GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B/W: 100Mb/s</td>
<td></td>
</tr>
<tr>
<td>Vehicles</td>
<td>4</td>
<td>CPU: 1core, 2.66GHz</td>
<td>Virtualized</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEM: 4GB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B/W: 100Mb/s</td>
<td></td>
</tr>
<tr>
<td>vDLT storage</td>
<td>12</td>
<td>CPU: 1core, 2.66GHz</td>
<td>Virtualized</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MEM: 16GB</td>
<td>(vDLT producers)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B/W: 100Mb/s</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Configuration of components for simulation.

The virtualized components in Table 5.1 mostly are built via containerization based on Docker. The communication among virtualized components is supported by an overlay network which is built by integrating Weave Net [130] with Docker. To enable VANETs, and use the AODV routing protocol, a P2P networking service is developed in the virtualized VANET nodes, which interacts with NS3 [96] to periodically generate the routing tables of AODV, and accepts the routing tables to guide their connection with each other. By using NS3 to generate the topology of VANETs having the same number of nodes, a series of AODV routing tables are calculated. To use the routes in
application instead of in kernel space, a series of matrices are created to map with the routing tables. An example of the matrices at a time point is given in Matrix 5.4.

\[
\begin{bmatrix}
RS1 & RS2 & TL1 & TL2 & VC1 & PD1 & PD2 \\
RS1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
RS2 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
TL1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
TL2 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
VC1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
PD1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
PD2 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\end{bmatrix}
\]

(5.4)

According to Matrix 5.4, connection between two virtualized nodes is denoted as 1 and disconnection as 0. With the names of nodes in the matrix are replaced with the IP addresses, the effective routes are clarified. A range of these matrices collaboratively reflect the AODV routing policies in a period of time, and hence support our VANET topology. Meanwhile, the firewall policy in operating system is used to conduct the flow of data according to routing tables. In our project, Iptables [119] is the firewall used, and according to the variation of matrices and time slots, we write a set of automation scripts to change the rules in Iptables forwarding, input and output chains to make the data flow consistent with the AODV routing tables. In vDLT, 12 producers are enabled to alternatively take control of producing blocks, and the producers also act as validators and feeders. Each block production in our experiment takes 500 milliseconds, and each block confirmation requires approvals from 5 of producers or validators.
5.3.2 Evaluation of Results

First evaluation is performed to check the effectiveness of uploading and downloading visual traffic events. The evaluation result is shown in Fig. 5.7. By comparing the videos which are respectively uploaded and downloaded, and the watermarks which are shown in both directions, it can verify the functions of sharing traffic events via BC together with the frame verification and watermarking utilization.

In Fig. 5.7, the uploading process is shown at the top, while downloading at the bottom. It is obvious that, the picture frames downloaded from vDLT keep the same as the ones uploaded, except the watermarks which BCAPs add the trust rate based on the frame verification to vDLT and embed it to the video in downloading procedures. Besides, from the result, the time scaled in milliseconds and location of the traffic are embedded in the video before uploading, which helps tell the vehicle movement. In this scenarios, time watermark in the result pictures implies that, a vehicle is driven without yielding for potential pedestrians when coming to a crosswalk, and hence someone can report this impolite driving behavior to improve the traffic safety around there. The watermark of trust rate indicates that some frames in the video are lost. The issue might be the result of unreliable network condition, malicious vandalization or attacks, and thus the trust rate is used to warn the downloader with caution. In general, the result demonstrates the feasibility and potential of our system in terms of improving traffic safety.

Furthermore, the efficiency of our system is evaluated mainly based on latency, throughput and the performance of frame block production. The evaluation focuses on the uploading process, since downloading and uploading processes work at the similar patterns, which experience multimedia transmission and processing in CLIs and BCAPs, which means that the uploading performance can reflect the other one. Besides, the uploading procedures determine the performance of conserving video frames in BC, and accordingly the quality of videos for downloading, therefore their
evaluation gains the most attention. The followings proceed to the detailed analysis with figures.

Fig. 5.8 records the evaluation result of throughput. To clarify size of data volume and its impacts on BC storage, it is necessary to analyze throughput of multimedia transfer. This evaluation reflects the data volume being saved in BC, which consequently affects the performance of data synchronization and consensus.

As seen from Fig. 5.8, the beginning stage experiences the boost of data transfer, and the value peak around 9 KB/s. In this stage, CLIs and BCAPs need to exchange massive information to create the connection, including multimedia codec and RTP contexts. But meanwhile, the cameras keep capturing the traffic events and wait for success connection to send the frames, and hence with connection built a large number of frames can be sent immediately. Obviously, the throughput in that time is seen surging. After the initial stage, the throughput reaches lower peaks in about every 8
seconds, which are commonly contributed by the I-Frame transfer. In our experiment, GOP length used in multimedia is set as 250, which means I-frame appears every 250 frames, and together with FPS set to be 30, therefore, the I-frame repeats in every 8 seconds approximately. Other than those moments, mostly, throughput hovers around 0.5 KB/s, because the B-Frame and P-Frame which are significantly smaller than I-Frame are actually the most frequently generated and transferred. Besides, frames are transferred in the compressed format which results in the average low throughput. Therefore, the frame data can be safely and conveniently wrapped into a transaction without complicated segmentation or compression techniques.

Fig. 5.9 compares the latency generated in BCAPs and vDLT for uploading and preserving the frames of traffic events. The delay to receive TINFOs in BCAPs is evaluated only for remote CLIs, while the latency of preparing in BCAPs shows the time on converting TINFOs from all CLIs to TTRXs. For vDLT, the latency is mainly contributed by producing blocks for multimedia frames. According to Fig.
Figure 5.9: The time latency evaluated in BC access points and vDLT

5.9, the delivery of traffic event frames from CLIs to BCAPs encounters the highest latency, and thus preparation of uploading frames in BCAPs and block production of frames in vDLT accordingly are delayed. The dramatic latency in this phase is caused by the initialization of RTP unicast, in which CLIs and BCAPs exchange much information about RTP and codec contexts to create the related stream, and during this time, frames captured in CLIs are delayed for transfer. Therefore, once the connection is created, all the frames buffered in CLIs are poured into BCAPs, but after that the transfer of multimedia becomes steady and has the latency consistent with timestamp increment which is 30 milliseconds as related to FPS. Accordingly, BCAPs see the highest delay of the beginning TINFOs, but according to the figures of preparation delay, BCAPs spend less than 50 to convert the TINFOs to TTRXs, which is the distance away from the line of receiving delay. In other words, the efficiency of BCAPs supports the frames in a time interval of block production to catch the current block. Also due to the dramatic transfer delay, frames from CLIs at the beginning
Figure 5.10: The efficiency of producing frame blocks in vDLT

are wrapped into blocks after a long wait. But soon after the upheaval, frames are steadily written into blocks.

Fig. 5.10 illustrates the performance of block production. As aforementioned, frames which miss being wrapped into the current block need to wait for the later ones. In other words, these frames are delayed to achieve consensus and preservation, and correspondingly reduce the efficiency of retrieval by clients, therefore this block production performance critically affects the entire system. As seen from Fig. 5.10, at the beginning when massive frames are transferred, the blocks produced hold the most number of frames. This situation is consistent with the analysis above, as immediately after CLIs and BCAPs are successfully connected, a large number of TINFOs burst into vDLT through BCAPs. Together with the dynamic adjustment of frames in transactions, it is reasonable to see vDLT producers wrap up almost 90 frames in a block. This number shows that there are 3 seconds of frames to be distributed over vDLT in 500 milliseconds which is the time of a block production. Furthermore, with
the approximately 10 blocks at the beginning able to handle all the deferred TINFOs, and the subsequent performance of producing frame block become steady. Generally, in the steady stage, two blocks can carry 30 TINFOs which is value of FPS. Since producing two blocks takes 1 second, the 30 frames per second can be timely uploaded into BC without being missed, and then with collaborative distributed consensus and storage, the prompt and reliable sharing of traffic information is achieved. From this result, it can also be seen that, the efficiency of BCAPs is remarkable, as they promptly convert the TINFOs and upload the TTRXs to vDLT for timely block production.

In summary, our experiment incorporates various performance evaluations to clearly demonstrate the feasibility and efficiency of proposal. The downloading and uploading processes are conducted to evaluate the system feasibility, while efficiency evaluation focuses on the uploading procedures. Since both directions work at the similar patterns, which experience multimedia transmission over network and multimedia processing inside CLIs and BCAPs, therefore, the uploading efficiency can reflect the other one. Moreover, to save and secure multimedia with BC encounters more challenges and urges enhanced improvement, therefore it is necessary to dig deeper in the multimedia transfer to BC.

5.4 Remarks

A system that integrates VANETs with BC technology for reliable and efficient sharing of visual traffic information is proposed to improve the traffic safety in this paper. Traffic safety has been an crucial issue, and hence increasingly gains research and engineering attention. As encouraged by the immutability of BC, we developed a system that employs BC technology to share the trustworthy visual traffic information in VANETs for safety improvement. However, BC commonly sees challenges while processing large volume of data due to its insufficient scalability.

To this end, we exploited the frame feature of multimedia, and use vDLT as the
BC platform, to improve both reliability and efficiency. vDLT as a service-oriented BC system with DPoS consensus algorithm and virtualization supports, is productive and efficient for BC development and deployment. In order to enable deeper multimedia protection and more efficient multimedia transfer in BC, we processed multimedia of traffic information at the frame level. To support the transfer of multimedia frames via BC, a 3-tires model is designed to clarify the relationship among the relevant components. In the model, the top layer is the data for end clients, the bottom one represents the block storage inside BC, and the middle bridges two end layers by providing conversion between frames and transactions and access points to BC. With the extraction of compressed multimedia frames, the transaction is competent at carrying multiple frames and hence the whole stream of multimedia, and hence the feasibility and efficiency of multimedia sharing via BC is achieve. To improve reliability, our system provides the verification of frame integrity based on the RTP timestamp and sequence, together with watermarking to protect the frame content. The increment of RTP timestamps is constant and thus the variation of increment implies the data loss or malicious vandalization on the multimedia traffic information. Watermarking is used to prove the trust of traffic information, since it is difficult to modify, and the change of frame can cause inconsistent watermarks. For higher efficiency and reliability, a dynamic adjustment of frames in a transaction is developed, by which transactions can carry the most frames to their best efforts and hence to reduce the latency of uploading frames into BC. The development of system involves the client applications, a range of services in vDLT, and a set of collaborative SCs. Techniques of multi-threading and message queuing are applied to achieve higher concurrency for better performance. SCs are developed based on modularization which can facilitate the agile extension of system and the swift migration of functions.

The result of our experiment demonstrates the advantages of our system. First of all, the feasibility is achieved, with the watermarks properly protecting traffic events, especially with the trust rate displayed in the multimedia downloaded to further reveal
the reference value of report. Moreover, latency, throughput, and block production performance are jointly evaluated to prove the efficiency of system. From those results, our system is able to timely transfer frames from clients to BC, and hence promptly share the trustworthy traffic information over the VANETs.
Chapter 6: DDL Based on vDLT with 6G Features

The 6G network targets the Internet of everything (IoE) implementation, and DDL can promote this progress with innovative performance in generating intelligence. Meanwhile, the 6G networks support ultra-reliable and low-latency communication (uRLLC) and thus can further elevate the DDL performance to empower the IoE development. However, DDL designs mostly focus on individual areas and yield separate intelligence which is insufficient for the IoE; besides, DDL platforms are usually managed in the centralized fashion, which is vulnerable for data preservation and task execution; the complexity of 6G networks involving heterogeneous devices and relations aggravates issues about reliability and efficiency of DDL. To this end, we propose a novel BC-escorted 6G-based DDL design for trustworthy model training. In this system, the 6G network design is utilized for system-wide uRLLC; non-homogeneous edge devices are grouped up with weighted consideration for DDL to train CNN models; MBSs and SBSs jointly provide two-tiers parameter aggregation to elevate the knowledge level; a dual-driven BC consensus is designed to verify tasks and models; users anywhere can retrieve models via the BC nodes for object detection. The proposed design is evaluated in comparison with Cloud-based and P2P-based DDLs, and the results demonstrate better performance on accuracy and latency achieved in the proposed system.

6.1 System Design

The focus of analyzing the system design is put on the network architecture and the model aggregation process. BC is used to provide a platform which secures the parameter sharing and the task distribution, and hence the communication via SCs and block synchronization need to be in-depth analyzed.
Figure 6.1: The overall architecture of proposed system in which the highlighted MBS is currently creating blocks of aggregated models.

6.1.1 Architecture

The proposed architecture targets the 6G network which incorporates heterogeneous edge devices, SBSs, MBSs, users and the InterPlanetary File System (IPFS) [11]. Fig. 6.1 illustrates the overall architecture of proposal. As shown in Fig. 6.1, IPFS stores the training dataset, the BC network supports data sharing via transactions and blocks, the 6G design provides MBSs and SBSs with uRLLC as the underlying communication support, and heterogeneous edge devices are connected with SBSs to form DDL platforms.

In particular, the address of dataset in IPFS is preserved in BC ledgers, and the modification of dataset will cause the change of address and the according sequence of blocks; thus, the consensus on the addresses of tampered dataset among BC participants cannot be approved. In other words, dataset in IPFS is protected and tamper-proof via the BC system. The unconfirmed dataset addresses cannot be
preserved in BC ledgers, such that the retrieval of dataset via BC is trustworthy.

Generally, in the proposed system, an entire model is produced by aggregating multiple sub-models in MBSs, while the sub-models are trained by SBSs, and the BC consensus continuously and comprehensively protects the trust of training dataset, parameters and tasks to escort the whole DDL process. The major components and their features are explained as follows.

1. A DDL platform is formed by one SBS and multiple heterogeneous edge devices, and the DDL functionality is developed based on lightweight CC platforms, which can be agilely achieved by tailoring Spark, K8S, etc. DDL platforms train sub-models using small-size dataset which is directly obtained from IPFS using the unique address received from BC transactions, and hence can improve both performances in getting dataset and sharing parameters, especially within 6G networks. Sub-models are transmitted via BC transactions by SBSs to BC producers.

2. The group of edge devices may incorporate vehicular system, computers, servers, smart phones, etc, and hence they have different resources equipped. The DDL scheme in this design considers the resource levels of edge devices and accordingly the SBSs for training task distribution and dataset allocation. The purpose of this weighted consideration is to elevate the DDL performance and to achieve the balanced parallelism of training.

3. SBSs bridge the model sharing between DDL platforms and the BC system. At one end, they are the controller of managing DDL, and responsible for task distribution based on the statuses of connected edge devices, as well as for updating parameters with edge workers. On the other side, SBSs synchronize blocks from BC system, and are triggered by the transaction within blocks for DDL training. The heterogeneous devices are vulnerable due to resource
limitation, such that it is reasonable to have BC system to verify the trust of sub-models.

4. MBSs are the powerful participants of BC system, which focus on block production, block consensus, and sub-model aggregation. To verify the trust of sub-models, MBSs collect the conditions of SBSs and their DDL platforms, in order to properly arrange training tasks. MBSs deal with aggregating sub-models, but without dataset, and this scheme is inspired by Federate Deep Learning (FDL) which considers the SGDs of all the workers. Communication between MBSs can go over the backbone network, such that the latency of block sharing is reduced dramatically.

5. Dynamic connection is supported in our BC system, which means that extra devices or equipment can join the BC networks as block producers, block validators or BC access points, and this feature allows for scalability improvement. However, to become the major BC nodes, devices need to be verified for consensus among a group of BC witness nodes. As the range of BC network can be flexibly extended by incorporating more participants, users or clients anywhere can retrieve the model from closer BC nodes.

In general, this layered design of DDL facilitates scalability improvement. With the femtocell support in 6G networks, incessant SBSs can be created, and various devices can faster locate and join the nearby SBS, such that DDL platforms designed in this architecture can be promptly empowered and spread. Therefore, plus the MBS to protect the data with great power and integrity, both the capability and security of DDL in this system can be significantly improved.
6.1.2 Model Aggregation

An entire model generated in the system requires the aggregation of multiple sub-models in MBSs, which are trained by SBSs. The process to create the final model is activated by the task assignment transaction which is created by requesters and subsequently broadcasted via blocks, and is finished by the model aggregation which finds the model of minimized loss.

To allow optimal task assignment, MBSs require to periodically collect the running condition of SBSs. The status of SBS is reported via transactions, which is sent from SBSs to current MBSs. The proposed system adopts incremental method in
transmitting and storing the status of SBSs, such that the latency and the resource utilization in this periodical communication can be improved.

Moreover, in a SBS, a publish-subscribe channel is developed to support the interaction of its BC system with its outer application. The DDL controller in a SBS subscribes the training trigger message, and once the SBS receives blocks of its responsible training task, it publishes the event into the channel. Subsequently, the DDL system obtains information about task, model and dataset in local BC ledgers, and group up edge devices to start out training.

The general process about model aggregation is depicted in Fig. 6.2, which focuses on the flows of blocks. According to Fig. 6.2, transactions of sub-models can reside in multiple blocks because the time of training varies in different DDL platforms, and the integrity of blocks is protected by their sequence and signatures. The major steps in this process are analyzed below.

1. A requester for models creates a transaction about the task and dependent dataset information. The transaction is broadcasted to the current block producer. It means that this trigger will be sent to the qualified SBSs, which are evaluated by MBSs in advance. The statuses of MBSs are saved in BC networks, and the current producer obtains the status records followed by deciding a set of SBSs for DDL training. The block carrying task assignment transactions is broadcasted over BC networks. After the transmission time, the block is delivered into the designated SBSs, then inside the block SBSs can find their task assignment, and hence SBSs are triggered to call the edge devices within DDL platforms for sub-model training.

2. For improved performance, SBSs analyze their group of edge devices and split the task to smaller portions which are distributed to workers. During DDL, the group of edge devices firstly retrieve the required dataset slices from IPFS using the addresses which are saved in the task assignment transaction; then
start out training, and continuously send their parameter updates to the related managing SBS, followed by getting the updated global parameters from SBS for training adjustment. Besides, the SBS record the performance and parameter contribution of each edge worker into the model transaction. After achieving the minimized loss, the SBS finalizes the transaction with a decided set of parameters, and send the transaction to BC networks.

3. The transactions of sub-models are delivered to the current block producer, through a series of hops from the SBS senders to the MBS producer. Once the MBS producer gets the transactions, it starts to verify the signature by using the relevant public key. The signature is created based on the transaction content and time information, such that the failure in checking signature means the transaction is malicious. The passed transaction will be parsed for getting its actions, which indicate the task distribution and the parameter contribution over the edge devices within a DDL platform, and hence these small tasks can be verified by their comparison with the task assignment for the SBS. As actions record the parameter contribution, it is available to verify the sub-model in terms of parameter information. Parameters and tasks are connected with each other, and changes on either side will affect the other one.

4. After the transaction count is successfully matched with the task number, and the verification of both parameters and tasks is passed, the current block producer starts the model aggregation. This is one round of training, and the resulted model is saved in BC distributed storage. If the quality of model is lower than the expected level, the MBS will build a new transaction about task assignment, and send it to the previous working SBSs. This indicates the beginning of the second training round. After a certain number of rounds, the expected loss level is achieved, and the producer stops assigning tasks, which means that the training on the specific dataset is finished, and relevant participants can use the
Therefore, the training performance and model quality is determined by the SBS DDL platforms. The status of edge devices and SBSs provides the necessary information for premium task distribution. With the 6G network support and the small size of dataset slice, the communication for parameter updates and data retrieval within DDL platforms can be dramatically expedited. This analysis supports that the proposed system can accelerate the training and meanwhile provide trust and privacy protection to the training tasks.
6.1.3 Smart Contracts

There are three major SCs used for transaction creation, and they are inter-connected to support inline execution. Fig. 6.3 shows the three SC structures and their relations. As shown in Fig. 6.3, the SCs for task assignment and for parameter sharing are related through mapping fields of task name and model name, and SCs for inline support of exchanging basic information, such as registration, dataset global address, keys, account events, etc, is used for verification of training and accounts. The inline function is implemented by exploring the in-memory SC actions which indicate the addresses of relevant functions and data structures. Essentially, inline support is the execution across SCs inside BC system.

Since the task and parameter sets are connected on one-to-one relation, the verification on tasks or on parameters triggers the same procedure to either side, such that dual-driven consensus can be developed. Moreover, for verifying the trust of tasks and parameters, the related SCs store the necessary fields, such as task ID and parameter ID. The base SC, which provides miscellaneous information, serves as the registry office for roaming among SCs, and before consensus, transaction creation obtains the required data and gets facilitated, which means that most of important information is available in the table without traversing the block-chain. To further promote the data query, SCs support the primary key and indexing of records, and BC nodes can create the record views for fast data selection.

6.2 Problem Formulation

As the design incorporates intensive communication among the BC nodes and DDL participants, it is necessary to analyze the latency. Meanwhile, the model aggregation is analyzed for the quality of model improvement. The major expectation is to find the accurate model with short time expense.
6.2.1 System-Wide Latency

The system-wide latency incorporates the time on creating blocks, DDL training in SBSs and the communication of blocks and transactions of models. In the proposed system, the consensus algorithm is designed based on DPoS-PBFT, and the interval of creating blocks is a constant value. Besides, the model aggregation is collaboratively processed by MBSs during block creation. To reduce the latency for producing a high-quality model, that is to accomplish the model aggregation in as few as possible blocks.

For one-round training which starts with getting task assignment from blocks in SBSs, and ends by aggregating SBS models to blocks in MBSs, the total time is denoted as $T_{\text{round}}$.

$$T_{\text{round}} = \max(T_{\text{sb}} + T_{\text{st}} + T_{\text{sm}} + T_{\text{ma}}, T_{\text{mb}}) \quad (6.1)$$

Where $T_{\text{sb}}$ pertains to the time of receiving the block in SBS, $T_{\text{st}}$ to the DDL training time in SBS, $T_{\text{sm}}$ to the time of returning model from SBS to current block-creating MBS, $T_{\text{ma}}$ to the time of model aggregation in MBS after receiving model transactions from SBSs, and $T_{\text{mb}}$ to the time of creating blocks which carry all the small models from SBSs. Generally, to achieve the minimal latency, one-round training needs to be accomplished in less than one interval of block production which is a constant value.

For communication about the block broadcast and transaction delivery, taking $n$ hops between current block-creating MBS and the targeted DDL SBS, the time for transmitting a block or transaction is defined as $T_{\text{sb-}n}$.

$$T_{\text{sb-}n} = \sum_{i=1}^{n} \left( \frac{D_{s}}{R_{i,i+1}} \right) \quad (6.2)$$

Where $R_{i,i+1}$ means the data rate of channel between two BC node $i$ and $i + 1$, and
$D_s$ means the block size or transaction size.

$$R_{i,i+1} = B_{i,i+1} \times \log_2 \left( \frac{1 + P_{i,i+1}}{N\sigma^2} \right)$$ (6.3)

Where $B_{i,i+1}$ and $P_{i,i+1}$ is collected from the channel between two neighboring nodes.

In our system, both blocks and transactions are allowed to carry unlimited size of data. Apparently, the support of 6G uRLLC facilitates the dramatic reduction of communication time, and thus gives chances to promote the sharing of large-volume blocks or transactions in the BC networks; consequently, the parameter sharing and model aggregation gain efficiency improvement.

In terms of the training latency in a SBS-DDL group, the overall training time $T_{st}$ is decided by the number and capability of the edge devices. While there are $K$ edge devices with a SBS to form a DDL platform, the $T_{st}$ is determined by the slowest edge device.

$$T_{st} = \max(T_{k,s}), k \in (1, K)$$ (6.4)

Where $T_{k,s}$ means the training time of edge device $k$. In particular, this training process may comprise multiple epochs, each of which involves multiple iterations.

$$T_{k,s} = t_{k,s,d} + \sum_{e=1}^{E} (t_{k,s,e})$$ (6.5)

Where $t_{k,s,d}$ and $t_{k,s,e}$ respectively indicate the time of getting data slice from SBS $s$ to edge device $k$, and the time of training in edge device $k$; besides, $t_{k,s,e}$ includes the time of exchanging parameters and local training. Especially, $t_{k,s,d}$ can be zero if the edge device $k$ preserves the required data slice. Therefore, both $t_{k,s,d}$ and $t_{k,s,e}$ are determined by the condition of channel between edge device $k$ and SBS $s$, and the sizes of training data or parameters in exchange.

Accordingly, since SBSs assign various volumes of data to edge devices according to their computing capability or resource configuration, we define the capability of
edge device $k$ as $C_k$, which is decided by its number of CPUs $U_k$ and channel data rate $R_{k,s}$. Further, based on the resource circumstances of edge device $k$, we allocate the according size of data $D_k$ for its local training.

$$D_k = \frac{\alpha * U_k + \beta * R_{k,s}}{\sum_{k=1}^{K} (\alpha * U_k + \beta * R_{k,s})} * D_s$$

(6.6)

Where, $D_s$ indicates the data size for SBS $s$, and accordingly it is determined by the its overall training capability, which is contributed jointly by its internal $K$ edge devices; besides, $\alpha$ and $\beta$ respectively indicate the weights of CPU and channel influences.

$$D_s = \frac{\sum_{k=1}^{K} \alpha * U_k + \beta * R_{k,s}}{\sum_{s=1}^{S} \sum_{k=1}^{K} (\alpha * U_k + \beta * R_{k,s})} * D$$

(6.7)

Where $D$ is the total size of data set for the collaborative DDL process. In general, the statuses of edge devices determine the distribution scheme of SBS, and whenever edge devices connect to a SBS, their resource statistics including the variation can be obtained and analyzed by the SBS.

### 6.2.2 Aggregated-Model Accuracy

An aggregated model is generated by fusing the small models which are contributed by multiple SBS-centric DDL platforms in one round of training. The overall loss of one-round training is determined by all the results of SBSs.

$$F_r(W) := \sum_{D_s \in S} \left( \frac{D_s}{D} * F_s(W_s) \right)$$

(6.8)

Where $F_r(W)$ is the overall loss result in one round, and $F_s(W_s)$ is the loss of DDL in the SBS $s$, which is associated with a parameter set $W_s$ and data $D_s$. We consider the impact of a SBS DDL platform in terms of its capability; as the more powerful platform is given more data and tasks, its performance has greater impact in overall
results.

To train a desired entire model, that is to find an aggregated model $W^*$ which yields the minimal loss and weight update after a series of training rounds.

$$W^* := \arg \min F_r(W)$$ (6.9)

According to Eq. 6.8, the SDG trends of SBSs jointly but with different individual impacts, affect the overall SDG variation.

$$\frac{\partial F_r(W)}{\partial W} := \sum_{D_s \in D} \frac{D_s}{D} * \frac{\partial F_s(W_s)}{\partial W_s}$$ (6.10)

Thus the desired model means the overall SGD trend approaches stable, i.e., $\frac{\partial F_r(W^*)}{\partial W^*} \approx 0$. The model aggregation is processed based on the weighted summation of parameters.

$$W := \sum_{s=1}^{S} \frac{D_s}{D} * W_s$$ (6.11)

Moreover, in a SBS-centric DDL, edge devices update their local parameters in iterations, and exchange their parameters with SBSs in epochs. The parameter aggregation in SBSs is processed based on the updates of all the connected edge devices in epochs.

$$W^p_s = W^{p-1}_s - \sum_{k=1}^{K} \frac{D_k}{D_s} * (W^p_k - W^{p-1}_s)$$ (6.12)

Where in epoch $p$, the parameters in SBS $s$ are generated according to the updates of all edge devices which are based on parameters in the previous epoch. After receiving the parameters $W^p_s$ from SBS $s$ and epoch $p$, the edge device $k$ continues training with the updated, and in iterations, its local parameters are changed according to the loss
trend $\nabla L(d_{k,i}, W^p_{k,i})$.

$$W^p_{k,i+1} = W^p_{k,i} - \eta \cdot \frac{d_{k,i}}{D_k} \cdot \sum \nabla L(d_{k,i}, W^p_{k,i})$$ (6.13)

In general, for SBSs and edge devices, the loss function can be defined as referred to Eq. 6.8, i.e., $F_s(W_s)$ is defined as $\sum D_k \cdot F_k(W_k)$, and $F_k(W_k)$ as $\sum d_{k,i} \cdot f_{k,i}(w_{k,i})$. Together with Eq. 6.9, it can be found that, the SGD trend is spread starting from the edge device level and ending by MBSs. Therefore, the protection and performance improvement should start with edge devices, but also be extended to SBSs and MBSs.

### 6.2.3 Aggregation Process

As the model aggregation is processed within block production, we focus on the procedures of triggering DDL in SBSs, and collecting SBS models for their aggregation during creating blocks. Algorithm 2 explains the major steps from starting training task to fusing parameters.

To invoke SBS DDL system, the current block producer firstly acquires the distribution and resource statistics of SBSs, then creates a task with related dataset information, such as dataset storage address and password, and lastly splits the task into a set of sub-tasks together with related dataset slicing. The splitting results are used to create a transaction using SCs of task assignment, and the transaction indicates the expected score of aggregate model. Along with block creation, transactions are wrapped in the current block, followed by being broadcasted during block synchronization.

Designated SBSs will be triggered to train the desired dataset slice after receiving the block. After model training, SBSs fill in a transaction which is constructed by the model SCs with the final parameters, task distribution and parameter contribution of DDL edge devices. Finally, SBSs send the transactions to the block producer. While creating the next block, the producer has to parse the previous block, such that
Algorithm 2: BC-triggered-and-aggregated DDL

**Input:** Task-Assignment (TA), Dataset (DS), Expected-Score (ES), Smart Contracts (SCs)

**Output:** TA-Block, AM-Block

**Initialize:** find SBSs, split TA & DS to be \# parts, fill TA \( TRX \) with TA-SC, set \( Pars\{\} \)

**Procedures:**
while \( Block \) Production do
  generate Block \( ID \) → \( BID \)
  for elapsed Time < \( T_{block} \) do
    find \( TRX \) according to Model-SC in Block\( BID \)
    parse \( TRX \) with Model-SC
    if \( TRX_i \) matched TA & DS & \( i < \#TRX \) then
      if \( \# \) of Actions matched \( \#TA_s \) & \( \#DS_s \) then
        for each Action \( j \) do
          check nounce with Action content verify Action content
        end
        add Action \( j \) → \( Pars \)
      end
    end
    find \( TRX_j \) in Blocks → \( Pars \)
    parse SGD\( s \) → \( ES' \)
    if \( ES' < ES \) then
      fill \( TRX \) with TA-SC & Block \( ID \)
      sign and broadcast Block
    end
    else
      aggregate records in \( Pars \) → \( TRX_m \) with Model-SC
    end
  end
the blocks are chained up along with sequence verification. In parsing the previous block, the transaction of SBS models is retrieved from actions, after passing the verification of the nonce which is generated based on the number, structure and the content of actions. The final parameters of a SBS are used for model aggregation. Meanwhile, according to the transaction, the producer needs to verify the total number by comparing the initial task assignment along with collecting the previous transactions of models.

To speed up the aggregation, there are two unique indexes to be created for searching related records, with one for indexing tasks and the other for parameters. Indexing method adopts SHA512 function to get hash values, which allows the long-run increase of records and swift query of tasks or parameters. Beside the unique index, the proposed system supports multi-index generation. Specifically, Eq. 6.14 explains the generation of unique index of task and parameter records.

\[
\begin{align*}
I_{task} &= SHA_{512}(T_{name}, R_{acct}, R_{time}, M_{SBS}) \\
I_{pars} &= SHA_{512}(P_{name}, S_{acct}, I_{task}) \\
&= SHA_{512}(P_{name}, S_{acct}, SHA_{512}(T_{name}, R_{acct}, R_{time}, M_{SBS}))
\end{align*}
\]

In Eq. 6.14, \(I_{task}\) and \(I_{pars}\) respectively stand for indexes of task and parameter records, \(T_{name}\) stands for task name, \(R_{acct}\) for requester account, \(R_{time}\) for request time, \(M_{SBS}\) is the SBS metadata set, and \(S_{acct}\) is actually the SBS account; each of these arguments can be used to index the related record. Since parameters are the results of task execution, their unique index incorporates the task uniqueness; hence, it is efficient to verify the validity of tasks according to the parameter results, and vice versa.

After all transaction are collected, model aggregation is started, and the score is evaluated by accumulating the SGD changes and comparing the result with the initial task assignment, which means that, if score is lower than the expectation, the block
producer will create a new task and notify SBSs to continue the training based on the current aggregated model. Continuously, producers can obtain different versions of aggregation results and SGDs, such that the BC system can get the aggregated model which yields the minimized loss. In general, the communication pertains to the block sharing, and the topology of BC networks impacts the communication performance, such that BC running over a P2P layout in a mesh network can provide more transmission paths for reliable data sharing.

Eventually, with sub-models being aggregated in the current block producer, the resulted blocks carrying transactions of intermediate parameters and eventual global parameters are broadcasted to witness nodes. Consensus in this proposal simplifies the DPoS mechanism, by employing witness nodes in consensus, where more than two third witnesses achieving consensus can finalize the blocks. Generally, the witness nodes receive the blocks, search the original task record according to the parameter transaction, correspondingly perform two-direction validation, and exchange their results for consensus verification.

### 6.3 Simulation and Evaluation

To evaluate the proposed system, we need to develop the dual-driven consensus algorithm, SCs, and the trigger interface for DDL training in vDLT system, as well as the DDL platform based on Spark. The 6G network is simulated through NS3, and is used to provide the parameters of communication, such as the channel data rate, BS power, distancing scheme, to estimate the impacts of 6G on our system. The major configuration for simulation is listed in Table 6.1.

The BC system is developed based on vDLT, which takes the DPoS concept to improve the consensus algorithm. In this simulation, the time interval of producing one block is set to be 1 second, and the total number of produces is 5, which means that at least 3 producers need to achieve consensus to confirm the blocks, and also means
<table>
<thead>
<tr>
<th>Name</th>
<th>Specs</th>
<th>Desc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>CPU(16 cores), RAM (128 GB)</td>
<td>4 physical</td>
</tr>
<tr>
<td>vDLT node</td>
<td>CPU(4 cores), RAM(8GB)</td>
<td>16 virtualized</td>
</tr>
<tr>
<td>MBS node</td>
<td>vDLT node specs</td>
<td>5 virtualized</td>
</tr>
<tr>
<td>SBS node</td>
<td>vDLT node specs</td>
<td>11 virtualized, each with 10 workers</td>
</tr>
<tr>
<td>edge/mobile device</td>
<td>CPU(2 cores), RAM(8GB)</td>
<td>up to 50 virtualized</td>
</tr>
<tr>
<td>Spark node</td>
<td>CPU(8 cores), RAM(16GB)</td>
<td>12 virtualized</td>
</tr>
</tbody>
</table>

Table 6.1: The list of major components configured.

that the loss of 2 produces can still maintain the trust of system and data. A table is used to indicate the connection between edge devices and SBSs, and an example of this connection scheme is shown in Table 6.2. Our simulation mostly applies the virtualized platform which is constructed using VMware vSphere, along with the GPU driven, network and storage virtualized for flexibly scaling the computation and communication environment.

<table>
<thead>
<tr>
<th></th>
<th>SBS₁</th>
<th>SBS₂</th>
<th>SBS₃</th>
<th>SBS₄</th>
<th>SBS₅</th>
<th>SBS₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>E₁</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E₂</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E₃</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E₄</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>E₅</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E₆</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.2: An example of simulated connection between SBSs and edge devices.

The evaluation is performed by comparing the performances of latency, loss, accuracy, block number and transaction volume for model aggregation. Fig. 6.4 shows the number of blocks required for aggregating an entire model, in terms of different numbers of SBS DDL platforms. In particular, the DDL process could take a time period which crosses multiple block production time slots; thus, it is necessary to
evaluate the actual blocks for carrying the parameters, and the number of blocks which have been produced during this DDL. It shows that, as the number of DDL platforms increases, less blocks are required to aggregate an entire model.

![Figure 6.4: The number of blocks for an aggregated model.](image)

Moreover, the transaction amount in aggregating sub-models is evaluated, and basically the results in Fig. 6.5 reflect the transmission throughput. From Fig. 6.5, it can be found that as the training epochs increase, which lead to the SGD converging into a final state, the parameters in sharing decrease. Particularly, the BC system requires periodical broadcast of node status and data synchronization to maintain the distributed ledger.

Latency evaluation result is illustrated in Fig. 6.6, which is about the time of training the ImageNet dataset using the CNN design. The latency of the proposed system is lower than Spark-based DDL and EasyDist which is a practical framework. In Fig. 6.6, it shows that as more nodes participate DDL, the training period is reduced. Because the beginning step needs to transmit a block, which comprises a
number of system transactions, when the number of SBS DDL platforms is small, the latency is higher than the Spark-based one. However, as more SBSs join for training, the latency decreases faster, and thus the proposed scheme is seen better.

The accuracy is related to the testing of model, in which we retrieve the model and use the test dataset. Accuracy comparison is shown in Fig. 6.7. According to Fig. 6.7, where \( N \) represents the number of SBSs and \( K \) means the number of nodes managed by each SBS, it can be found that, the accuracy trends have slight difference while the number of SBS DDL platforms changes. The accuracy is still affected by the number of DDLs, as the parameter updates and model aggregation impact each other, and the performance is related to the overall configuration of SBS platforms. But after sufficient rounds of training, the accuracy can achieve a high-enough level, which proves the feasibility of system.

The loss evaluation finds us the performance of intermediate training procedures. Fig. 6.8 shows the loss changes of our system in terms of different number of SBS
DDL platforms in use. The configuration of SBSs is consistent with the one in latency evaluation. As seen in Fig. 6.8, while the number of SBS DDL platforms increases, the starting loss goes lower. This is beneficial from the overall capability of training. In the evaluation, workers for all DDLs are equipped with same capability, such that the parameter changes have same impact, and the DDL platform of better capability yields lower loss.

Furthermore, the performance of proposed system is compared with two baselines, which are DDL applications developed using Spark-based DL4J and EasyDist frameworks. The comparison focuses on the DDL accuracy and loss performance metrics. Fig. 6.9 and Fig. 6.10 respectively demonstrate the accuracy comparison in terms of training and validation stages, while Fig. 6.11 and Fig. 6.12 compare the loss situation also in the two major stages.

From Fig. 6.9, it is shown that, our proposed system achieves higher accuracy
Figure 6.7: The accuracy of proposed system in terms of SBS-DDL provision.

Figure 6.8: The loss of proposed system in terms of SBS-DDL provision.
in training. This accuracy improvement is beneficial from the better granularity of data slicing and training tasks, since the status statistics of SBS platforms is periodically reported to BC system, the training tasks are designed and assigned more reasonably. While the nodes of lower resource provision just handle the lightweight training tasks, the challenges in computing parameters mostly are solved by the powerful participants. Therefore, the reasonable task distribution and dataset slicing can considerably improve the training performance.

In the validation stage, according to Fig. 6.10, all the three applications approach stable levels of accuracy. In comparison, our system arrives at the stable state earlier, which means the time expense is reduced along with avoiding unnecessary training steps. The scheduling according to SBS platform resource condition significantly helps reduce the resource consumption.

Lastly, we measure the accuracy between the three system designs. In Fig. 6.11, which is about the accuracy evaluation in training stages, it can be seen that, accuracy
Figure 6.10: Comparison of loss performance in validation stages.

Figure 6.11: Comparison of accuracy in training stages.
performance is consistent with the loss measurement. The proposed design can achieve better accuracy in the earlier phase, which means the training time can be further reduced by improving the task assignment and data slicing mechanism. On the other side, in Fig. 6.12, the accuracy performance evaluated in validation stages also proves the advantages of scheduling based on the condition of resource provisioning in improving the DDL process.

The simulation in this paper involves the security evaluation, in which the mimic of attacking the DDL controllers is achieved by changing the dataset storage and parameter records. In comparison, our vDLT BC can maintain the main chain of blocks, even though one storage has been changed. However, by changing the database of clustering management in Spark platform, the DDL process is immediately broken and unable to continue even after the database restored; because the intermediate parameters are cached in memory and the machine state has changed but unable to notify each other via the managing component, the global information becomes
inconsistent. Generally, using BC technology plus scheduling design to protect the DDL system is able to improve both reliability and efficiency performances.

6.4 Remarks

As the future 6G network will release a bunch of innovative features, such as ultra-fast data rate, mmWave, massive MIMO, etc., researches on using the 6G design for data-intensive applications are increasingly conducted. As a typically data-intensive and communication-centric application, DDL can be empowered by 6G networks due to the uRLLC support in communication. On the other side, The 6G networks can gain increasing intelligence, since DDL becomes a prompt training scheme by distributing the training tasks over a number of workers in parallel. DDL address the efficiency issues, but it is challenging for reliability improvement. Centralized management is traditionally required in DDL for collaborating the parameter and task exchanges among workers and controllers, but this is vulnerable for malicious attacks without advanced protection on the managing components. Thus, in this paper, we propose a BC-escorted DDL scheme in 6G networking environment. In our system, SBSs in 6G networks are responsible for DDL management and MBSs for upper-level model processing. A dual-driven consensus algorithm is designed to protect the tasks and parameters simultaneously, and accordingly protect the trust and security of SBS management. Besides, collaborative model aggregation is proposed which is performed in multiple blocks and training rounds, and between various MBSs to progressively validate and aggregate the parameters for eventually producing a trustworthy model. With the features of BC, the privacy of training task and dataset is protected, since the data is broadcasted via blocks without being sent directly to the receiver, and only the dedicated SBSs with security keys can parse the training context. Data synchronization and layered design for collaborative DDL jointly facilitate the scalability and reliability improvements, in which there are more chances to empower
the DDL capability by incentivizing more edge devices for training and more SBSs for sharing parameters and to improve the DDL reliability by using more MBSs for model aggregation and consensus. In the system evaluation, the performances of accuracy and latency demonstrate significant improvements over centralization-based DDL schemes. Therefore, the proposed system can provide a trustworthy and prompt DDL platform.
Chapter 7: ZTA for Sharding-scaled BC-based VANETs

Utilizing BC to secure VANETs gains remarkable prosperity, but meanwhile attracts unending attempts to spoil the protection capability in BC-based VANETs. ZTA is being promoted to overcome the drawbacks of prevalent perimeter-based security solutions, and especially engages continuous and dynamic protection on ever-growing networks. Since both BC and VANET allow trustless participants, security issues of individuals inside can be explored to attack the system, and the overall cybersecurity level of network continually changes due to the mobility of participants, protection policies should be adjusted in quick response. Generally, ZTA with adaptive and border-free protection concepts, is promising to improve the cybersecurity level of BC-based VANETs. Therefore, we propose a framework which integrates ZTA with scalable-BC-based VANET (sBC-V-ZTA) to enhance the protection of data sharing. The proposed system collects device features to swiftly calculate the security index, hierarchically decides ZTA policies with security index as the primary attribute to augment the trust of devices and the security of data, and enables BC sharding with enforced interoperable consensus to improve system-wide efficiency and scalability. The experiment results demonstrate that, as compared to the single-chain architecture constantly running specific access policies, the proposed system yields better performance in data sharing.

7.1 Framework Architecture

The proposed design follows the divide-and-conquer principle of BC technology and enables BC sharding and decentralized ZTA to improve system performance comprehensively. BC sharding is an on-chain solution working in the Layer 1 of BC system,
Figure 7.1: The general architecture of proposed system which enables ZTA and BC sharding to jointly improve security, decentralization and scalability of BC-based VANETs.

and its significance focuses on in-parallel processing of transactions along with high interoperability between main-chain and shards. In this proposal, there is a main-chain managing the sharding scheme and cross-shards communication; ZTA is designed into our system for security and trust achievements, since per-shard validation capability is reduced as the scope and number of participants shrinks; ZTA management and access control are decentralized by BC innate mechanism and are distributed across main-chain and shard-chains; security policies are used to prevent the unauthorized privileges and eliminate the unauthenticated devices so as to protect against malicious attacks to BC storage via data sharing.
7.1.1 Network Structure

The proposed system is built upon VANET topology where a main-chain and multiple shard-chains are formed by VANET-involved devices. The network architecture is illustrated in Fig. 7.1. According to Fig. 7.1, the VANET majorly incorporates vehicles, RSUs and BSs, multiple shards are designated in ranges and the main-chain coordinates cross-shards activities. RSU and BSs are equipped with ZTA and BC functionality to cooperatively manage the data sharing in VANET. The deployment of devices, including their roles and relations is explained as follows.

- The main-chain is formed by connecting BSs in different areas, and it can be extended by incorporating more infrastructures; different shards comprise various numbers of RSUs and Vehicles, and are constructed based on the device and traffic condition in areas; both BSs and RSUs are equipped with computing resource.

- The authorization of data and policy is individually handled by main-chain and shard-chains; the main-chain allows and controls the data flows from RSUs, the shard-chain manages shard-based data and vehicle accesses; designated ports and links are provided for various communication purposes which are highlighted in different colors.

- BSs and RSUs are the block producer (BP) candidates running ZTA engines, and relevant governance components are distributed into both shard-chain and main-chain; main-chain BSs are responsible for shard construction and synchronization across shards, while shard-chain RSUs for the shard-based activities.

- Security issues should be collected and analyzed in the global and fair-and-square fashion, and thus the generation and preservation of dataset for model training is assigned into main-chain; main-chain holds the global view of the VANET,
and it provides sharding scheme along with consensus and validation to protect the integrity among shards.

- Due to the capability issues, training models for security indexing which is used for defining policy is performed and continually optimized in main-chain with sufficient updated dataset; the trained outcomes are derived into shards, and the participants inside shards can apply models and policies in terms of device status inside.

- Since the authentication requirement changes according to the network condition of shard, it is more efficient to execute and adjust the security policy inside shards; the verification of device and data can be collaborated via main-chain for cross-shards data sharing which prevents the performance from falling back to the case of single chain.

The case of cross-shard movement of vehicles is also depicted in Fig. 7.1, where the red vehicle is leaving shard2 and entering shard1. Upon taking on the data sharing of this vehicle, shard1 sends transaction to main-chain to obtain the security and trust verification results of this vehicle from shard2. In general, the proposed structure facilitates extending shard numbers and the main-chain scope to improve system scalability, and meanwhile supports top-down management on the whole system as well as decentralized ZTA for enhanced protection on both security and trust.

### 7.1.2 Procedural Workflow

The entire system requires the cooperation between main-chain and shard-chains, along with their own management schemes for comprehensive protection. The major procedural components and their relations are illustrated in Fig. 7.2. As shown in Fig. 7.2, there are five major components in the whole process, which are Main-Chain Sharding, ZTA Controller, ZTA Actuator, BC Executor and Main-Chain
Coordinator. These components are jointly invoked to perform inter-shard and inner-shard management on data sharing, and their security and trust are ensured due to the underlying ZTA access control and BC consensus. The major components are explained below.

- **Main-Chain Sharding** is the initial procedure, and it is execution requires the authority to collect the road-traffic information in related areas, such as the density of vehicles and the number of RSUs. Area division and RSU selection for shard-based validation and consensus is performed, and results are preserved in main-chain.

- **ZTA Controller** deals with the automation of generating dataset and training models, and the administration of policies and attributes. The dataset features can be generated by simulation on Cytoscape and OpenC2 [79] and the real-time reports from VANET devices, and seamlessly transferred to the training process.

- **ZTA Actuator** handles the assessment of device security status based on reported features, and the access control with policy enforcement in data plane and policy decision in control plane. Policies verify the device authority in using SCs to upload transactions and synchronizing transactions and blocks.

- **BC Executor** is responsible for executing transaction along with further authority verification during block production, as well as eventual block consensus and validation. Besides keys and signatures outside SCs, related actions with parameters defined in SCs are invoked to verify the trust of device and its data.

- **Main-Chain Coordinator** is the component of handling the cross-shards verification of device trust. It supports paging shards, since main-chain carries the sharing scheme, and provides SCs to accept requests from source shards, and to notify the destination ones. Notification triggers calling SCs in destination to execute verification and return receipt to the source.
Access control in this process is accomplished jointly by the ZTA actuators and BC executor. Requests of data sharing firstly enter ZTA actuators which immediately execute the inner access control model to verify device authorization for related permission or privileges on requested resource. Subsequently, the utilization of resource which may cause data changes experiences decentralized validation in BC system. Eventually, request-associated data is decided whether to be preserved in BC storage.

With BC and sharding supports, our proposal provides an improved ZTA design based on the ZTA diagram promoted by NIST [112]. The ZTA policy administration is designed into the main-chain for stronger capability, the policy enforcement and decision-making is developed in both main-chain and shard-chain for higher efficiency and scalability. The management of ZTA data and control planes is decentralized.
and cooperated by BC and sharding supports. The least-privilege principle of ZTA is ensured which allows the uploads of device status features and the initial parameters, and privileges can be adjusted for higher permissions in transaction and block processing according to their updated features. Moreover, besides vulnerability and malware, extended device features, such as reachability, packets in transmission and operating system version, are incorporated since they certainly are security impacts, and sufficient attributes benefit the scalability and reliability of ZTA access control. Continuous feature collection and dataset generation is supported, such that with updated and revised features, model training can progressively be optimized to improve the system performance.

### 7.1.3 Smart Contracts

There are a set of SCs are required to support interaction between above components and the BC system and meanwhile to secure the operations on BC storage. Fig. 7.3
illustrates the major proposed SCs and their relationship. SCs are the components inside BC system, which define functions and data structures in relevant scopes. In our proposal, the design of SCs follows the high cohesion and low decoupling principles to support the agile inline calls and meanwhile to conceal the sensitive data fields from each other; non-relational database but with data indexing is used in BC system for higher computation and storage efficiency. According to Fig. 7.3, actions inside SCs can manipulate the owned data, and can be invoked external SCs for data access, such as query and modification; a series of SCs can jointly accomplish a complex of functions. The major proposed SCs are listed in what follows.

- **Shard-Plans SCs (SP-SCs)** are used for uploading the sharding scheme which are called by relevant authorities, thus their utilization require the authority permissions; they store the sensitive data, including the information of RSUs and shards, to prevent repeated sharding plans.

- **Model-Upload SCs (MU-SCs)** are designed for ZTA Controller to upload the trained models and related dataset information in main-chain; there are two tables respectively carrying models and dataset indexed by their own identities; authority permissions are needed in execution.

- **Open-Account SCs (OA-SCs)** are commonly used by devices, including vehicles and RSUs, to report their status features, and are executed by ZTA Actuator to register or update these devices into shard-chains or the main-chain; account name and device identity form the primary index.

- **Policy-Setup SCs (PS-SCs)** are called by OA-SCs to trigger ZTA Actuator to use the model in MU-SCs to calculate security index and to setup policies for devices; policy and device identities are indexed in the table in PS-SCs which can be referred to OA-SCs and MU-SCs via their query actions.

- **Policy-Check SCs (PC-SCs)** are the essential support in verifying the trust of
transaction, and called by other SCs. They call the query action in OA-SCs and PS-SCs to obtain the account and policy information, and verify the security and trust proof; for shards, if the device info is absent, the transaction will be forwarded to main-chain.

- Cross-Shard SCs (CS-SCs) is a complex of functions deployed in main-chain for main-chain and inter-shards management; they can be used by RSUs and main-chain parties. The action of notification is designed to coordinate inter-shards transactions, such as model information, inner-shard validator assignment and cross-shards verification; the table stores paths regarding request, notification and response.

Moreover, the design of SCs in this proposal aims at tightly coupling the shard-chain and main-chain to ensure their interoperability. Unlike the layer-2 scaling solutions which pertain to creating isolated small chains or ledgers, the sharding supports in this proposal is highly dependent of the management of main-chain over shard-chains using SCs. For example, with related SCs, main-chain plans the shards and votes their per-shard validators, and these activities require the consensus in main-chain; upon receiving voting results from main-chain, the BP candidates will be enabled or adjusted in a shard to carry out their consensus procedure. While in sending transaction from shard1 to shard2, the results of verification though calling PC-SCs in shard2 determine the validity of transaction in shard1, and main-chain also validates the cross-shards transactions using CS-SCs for both sides. Therefore, the proposed SCs are necessitated to support the system mechanism.

In summary, ZTA access control, BC sharding and BC consensus are integrated to support the proposed system mechanism, and OS-level applications and BC-level SCs jointly forge an ecosystem to elevate the system versatility and expansibility. ZTA is deployed as the front-line proctor in a decentralized and hierarchical fashion with SCs as the inner and protected executors, can provide trustworthy access control
and multifaceted system protection for data sharing. BC consensus penetrates the main-chain and shard-chains; coordination in main-chain is necessitated in sharding schemes and for cross-shards data sharing. SCs bridge the OS application and BC storage; the principle of designing components in both OS and BC levels is centered on the improvement of system efficiency and reliability.

7.2 System Development

The proposed system is a complex of functions, and the most important supportive development will be elaborated, which include the generation of security index, hierarchical access control, sharding scheme and consensus approaches. Besides, the prospective of performance optimization will be also provided. Generally, our development aims to achieve the balance between system-wide security, decentralization and efficiency.

7.2.1 Security Features

The features to describe a device security status can be designed into multiple categories, such as vulnerability, process, service, application, firewall rule and network interface. For better explainability, we consider device features as pixels in an image to figuratively describe the device security status; moreover, pictures as inputs can facilitate the design and development of classification-oriented or clustering-oriented deep learning algorithms especially at higher proficiency and reliability.

The representation of a device with security features $X$ can use a $N \times N$ RGB image, i.e., $X$ is placed in $N \times N$ matrix. The placement follows feature category assignment. Feature categories are assigned different ranges of color space to make the presentation more intuitive and meanwhile to indicate influence degrees. Each
color channel of feature $x$ is calculated according to its category:

$$C(x, m, M) = \lceil F(\min(m, M), \max(m, M)) \rceil$$

$$\min(m, M) = (M - m - 1) \times \frac{256}{M}$$

$$\max(m, M) = (M - m) \times \frac{256}{M} - 1$$

for $x \in X, m \in M$  \hfill (7.1)

where $m$ is the category order in $M$, and with color ranges generated by dividing 255 into $M$ portions, for category $m$, each color channel is set by taking an integer from the range. Since color green is the most sensitive, we use $g$ channel to represent feature category, and $r$ and $b$ which can be random values together stand for the feature keyword to generate distinct colors among groups. Examples of describing device status in images are shown in Fig. 7.4, where darker pixels represent the bigger concerns about security, the length of similar pixels equals the number of same-kind features, such as vulnerabilities, processes and ports.

It is significant to represent device security concerns in pictures. According to Fig. 7.4, the picture with larger area of rich colors reflects more features or options which can be exploited for attacks. Conversely, the device of a clean picture means it performs patches or upgrades. In particular, a device with more firewall rules implies carrying more security issues or more sensitive, and hence more attractive to attackers.

### 7.2.2 Security Indexing

The proposed security indexing approach is developed based on BC and deep autoencoder clustering (DAC) concepts with a focus on balancing system efficiency and reliability. Most security assessment or testing approaches count on burdensome computation, which are impractical for VANET devices owing to the resource limitation [67]. But still, features for security assessment can involve numerous concerns, and
various security requirements make it difficult to decide the specific ranking strategy. Therefore, to practically assess security levels within BC-based VANETs, we design a lightweight but reliable BC-assisted DAC-based security indexing algorithm which is shown in Algorithm 3.

In Algorithm 3, \textit{Trainig} firstly trains the deep autoencoder, and then calls \textit{Clustering} to group up similar devices. Deep autoencoder is trained to find two models $W_e^*$ and $W_d^*$ which yield the minimal loss between encoder inputs and decoder outputs along with iterative weight updates. The training objective can be defined as:

$$W_e^*, W_d^* := \arg \min L(X, F_d(F_e(W_e, X), W_d))$$  \hspace{1cm} (7.2)$$

where $L$ is loss calculation, $F_e$ and $F_d$ respectively are the encoding and decoding forward functions, and $X$ is the original input set. The training supports mini-batch, which takes $X_B$ as iterative input batch, updates $W_e$ and $W_d$ according to loss $L$ evaluation, and till finds the minimal loss and eventually yields $W_e^*$ and $W_d^*$.

\textit{Clustering} is used into the data $X_e$ which is encoded by $F_e$ with original $X$ and encoder model $W_e^*$. In clustering, a set of $K$ random centroids $C$ are initially selected from $X_e$, each record in $X_e$ is compared with $C$ to find its closest centroids, clusters
$S$ are formed and their centroids are updated, and eventually ideal $C$ and $S$ are determined if $C$ is unchanged. At the end of *Training*, the clusters $S$, centroids $C$ and autoencoder $W_e^*$ and $W_d^*$ will be uploaded to BC. The number of clusters $K$ can be scaled to improve the cluster similarity.

*Indexing* function provides prompt on-site index generation based on device reported features. The trained model for encoding $W_e^*$ and clusters centroids $C$ are retrieved from $BC$ firstly, then $W_e^*$ is used to learn the features $X_{test}$ reported by device and lastly $C$ is for indexing the device status by searching the centroid closest to the device encoded representation. Generally, the time complexity in *Indexing* can be estimated as:

$$T_{indexing} \approx O\left( \sum_{l \in L_{enc}} (N_l \times N_{l+1}) \right) + O(N_{L_{enc}} \times K)$$

$$\approx O(L_{enc} \times N \times N)$$

(7.3)

where $N \times N$ images are inputs, the $L_{enc}$ and $N_l$ are respectively the number of layers and the number of outputs in layer $l$ during encoding, and multi-threading support is ignored. The encoded output is compared with $K$ centroids to assign the device into related clusters.

To improve training accuracy, features reported by device in practice are also uploaded to main-chain as updated dataset records. *Training* and *Clustering* can run in continuous iterations, in each of which previous models is retrieved from BC first, and then updated models and clusters after training is uploaded to BC. Accordingly, main-chain and shard-chains can adjust the device security indices to support dynamic access control.
Algorithm 3: BC-assisted DAC-based security indexing

**Input:** $X, P, B, K$

**Output:** $W_e^*, W_d^*, C, S$

**Initial:** load $X$, $\emptyset \to S$, BC $\to \{W_e, W_d, C\}$

**def** Clustering($X$, $C$):

1. $C = \{C : -\text{Rand}(X, K)\}$
2. for $V$ iters $\&\& C_p \neq C$
   1. $C_{prev} \leftarrow C$, $S \leftarrow \emptyset$
   2. $\text{argmin}(||X_i - C_j||^2) \to S_j \cup X_i, j \in K, i \in ||X||$
   3. $C_j = \frac{\sum_i^{||S||} X_i^{||S||}}{||S||} \to C, j \in K$

**def** Training($X, W_e, W_d, C$):

1. $W_e, W_d = \{W_e, W_d : -\text{Rand}\}$
2. for $P$ epochs do
   1. for $||X||$ batches do
      1. $X_B = F_e(X_B, W_e)$, $Y_B = F_d(X_B, W_d)$
      2. $L(X_B, Y_B) \to W_{(e,d)} = W_{(e,d)} - \eta \frac{\partial L(X_B, Y_B)}{\partial W_{(e,d)}}$
   2. Clustering($F_e(X, W_e^*)$, $C$) $\to \{W_e^*, W_d^*, C, S\} \to \text{BC}$

**def** Indexing($X$):

1. $\{C, W_e^*, S\} \leftarrow \text{BC}$
2. $X_e = F_e(X, W_e^*) \to \text{BC}$
3. $\text{argmin}(||X_e - C_j||^2) \to \text{Index}(X_e, S) \to \text{BC}$

**Procedures:**

1. Training($X_{\text{train}}, W_e, W_d, C$) $\to \{W_e^*, W_d^*, C, S\} \to \text{BC}$
2. Device $\to X_{\text{test}} \to \text{Indexing}(X_{\text{test}})$
7.2.3 Access Control

The proposed access control is a hdPBAC model executed by BC participants and is developed in accordance with the proposed architecture. Fig. 7.5 describes the hdPBACs model where ZTA access control is associated with SCs to prevent untrusted data from entering BC preserved storage.

According to Fig. 7.5, hdPBAC inherits the PBAC essence, and increases methods to comply with the BC protocols. In hdPBAC, besides security index, device role is a major attributes, and device roles are designated in hierarchy which restricts the request access between devices. Policies are administrated by PAP which holds privileges of submitting policy-related information into BC system and bypassing PEP; submission experiences consensus in order get preserved in BC storage; users
Algorithm 4: Transaction sharing with hdPBAC

Input: \(REQ, \text{Wallet, Sessions}\)
Output: \(RET\)

Initial: \(\text{Sessions} \rightarrow \text{Session}, \emptyset \rightarrow \text{REQ, RET}\)

Procedures:
if USR send TRX via SC then
  \(\text{Wallet} \rightarrow \text{Role, Secu, Key} \rightarrow \text{Attrs}\)
  \(\text{REQ} = \{\text{USR, GET, \{T_{SC} | SC\}, \{Role, Secu\}}\}\)
  \(\text{RET} = \text{Session.send(REQ)}\)
  if RET.SC! = \emptyset then
    \(\text{TRX} = \text{RET.SC(\text{TRX}_{raw}).ES(Key)}\)
    \(\text{REQ} = \{\text{USR, PUT, }\)
    \(\{T_{TRX}[TRX]\},\)
    \(\{\text{Role, Secu, RSU}_{\text{dst}}\}\}\)
    \(\text{RET} = \text{Session.send(REQ)}\)
  end
end

if DEV recv REQ then
  \(\text{Wallet} \rightarrow \text{Role, Secu, Key, RSU} \rightarrow \text{Attrs}\)
  \(\text{BC} \rightarrow \{\text{Perm, Rule}\} \cup \text{Attrs}\)
  \(\text{REQ} = \text{Session.recv()}\)
  \(\text{RET} = \text{Rule(REQ.Attrs.Role, Role).Verify()}\)
  if REQ.Attrs.RSU_{\text{dst}}! = RSU then
    \(\text{REQ} = \{\text{RSU, FWD, REQ, }\{\text{Role, Secu}\}\}\)
    \(\text{Session.send(REQ) #} \rightarrow \text{Main/RSU}_{\text{dst}}\)
  end
else
  \(\text{RET} = \text{Rule(REQ.Attrs, Attrs).Verify()}\)
  \(\text{RET} = \text{BC(REQ, BC.get(SC, TRX)}\)
end

inquire \text{Wallet} for attributes to create request; in getting requests, \text{PEP} extract the attributes and call \text{PDP} to make decision; \text{PDP} retrieves information from BC in read-only mode and return decisions to \text{PEP}; requests can be relayed to BC resource area for permitted operations; eventually BC system validates request-generated data to decide its storage policy.

The development for sharing transactions follows Algorithm 4. Request \(REQ\) is
formatted as \{Account, Action, Object, Attributes\} where Attributes can be retrieved from Wallet. The information about Sessions is provided to indicate available access points. To send transaction TRX via action PUT, USR needs to get related SC, and encrypts and signs the TRX. When DEV receives REQ from Session, and uses the rule to verify Role. For example, visitors such as vehicles and mobile phones are permitted to access shard-chain RSUs but prohibited to enter main-chain servers; shard-chain RSUs are allowed to communicate with main-chain servers. DEV decides whether to forward via action FWD to destination. If DEV is destination, permission Perm related to security index Secu and extra Attributes retrieved from BC are used for policy decision. For block sharing, the related parameters need to be changed, and the procedures are similar to transaction sharing. In particular, if Secu is absent in Attributes, it means the USR has yet to report features for security indexing by using related SCs. When opening accounts in system, devices can get their own Attributes. The policies can be adjusted by main-chain authorities, and shard-chain participants will be notified to get updated policies and security assessment approaches.

### 7.2.4 Sharding Scheme

Sharding in this framework is initiated according to traffic volume and security indexing in areas. Considering the commute habits, the initiative of sharding is based on the geographic areas, and in each area the traffic and security situation is evaluated to form the initial BC consensus committee.

In shards, the higher reliability and efficiency of BC consensus can be achieved by employing the RSUs which are rarely busy but considerably secured. It is reasonable to consider the RSU popularity in terms of the RSU traffic volume, and choose the RSUs of lower popularity into the consensus committee. Moreover, the security ranking of RSUs in committee should meet specific requirements in order to become the validators or producers. Thus, the objective of initiating a shard can be defined as:
\[
\sum_{i \in N} \left[ (p_i < E_p) \land (s_i > E_s) \right] > M
\]  

(7.4)

where \( N \) is the total number of RSU in one area, \( p_i \) and \( s_i \) are respectively the popularity and security rankings of the \( i \)th RSU, \( E_p \) and \( E_s \) are the ceiling popularity and the ground security requirements for RSUs to join consensus committee, and the amount of qualified RSUs should larger than a threshold \( M \).

The RSU popularity in terms of its traffic volume is considered as following Zipf–Mandelbrot [77] distribution in one area, since the most busy RSUs consume the majority of traffic data. For an area equipped with \( N \) RSUs, the set of traffic volumes is denoted as \( V = \{ v_1, v_2, ..., v_N \} \) and the according popularity set \( P = \{ p_1, p_2, ..., p_N \} \), where \( v_i \) and \( p_i \) are respectively the traffic volume and popularity of the \( i \)th RSU. Based on Zipf–Mandelbrot, the popularity in terms of traffic volume for the \( i \)th RSU can be calculated as:

\[
p_i = \frac{(v_i + q)^{-\alpha}}{\sum_{v_j \in V} (v_j + q)^{-\alpha}}, \quad v_i \in V
\]

(7.5)

where \( \alpha \) and \( q \) are the coefficients jointly to scale the distribution, and for \( q = 0 \) the model becomes Zipf. Apparently, Zipf–Mandelbrot is more flexible to yield the expected popularity according to real situation. For the \( i \)th RSU, security ranking \( s_i \) can refer to its security indexing and can be calculated in terms of the size of centroid encoded image in its cluster \( C_{k,i} \), i.e, \( s_i = \sum_{j \in ||C_{k,i}||} g_j \) where \( j \) is the pixel index and \( g_j \) is the pixel value.

Sharding scheme is developed based on Algorithm 5. DPoS-PBFT is the consensus algorithm used in main-chain and shard-chains, in which DPoS is for voting the BPs and PBFT for validating and confirming blocks. As shown in Algorithm 5, shards are created and updated by main-chain participants and the results are sent to shards after main-chain Consensus. Initially, main-chain PB candidates \( \{BP_{mc} \} \) are voted
Algorithm 5: Sharding scheme upon creation and patching of devices

Input: $\alpha, q, E_p, E_s, A\{rsu : \{v, s, b\}\}, M, H, B, T_{out}$

Output: $\{SD\}$

Initial: $\{BP_{mc}\}, A \rightarrow Areas\{area{\{RSUs, SD\{\emptyset\}\}}\}, Timer$

Procedures:

$BC \rightarrow Areas$

for $area \in Areas$ do

if $area.RSUs == \tilde{\text{Areas}}.area.RSUs$ then

| continue $\rightarrow$ #SKIP : no update |

end

for $rsu \in area.RSUs$ do

| calculate $p_{rsu}, s_{rsu}$ #in Eq.7.5, Algo.3 |

| $(p_{rsu} < E_p) \& (s_{rsu} > E_s) \rightarrow rsu \cup area.SD$ |

end

if $(r = ||area.SD|| - M) < 0$ then

| $[\bar{rsu}] = [rsu \text{ for } (p_{rsu} < E_p) \& (\downarrow r > 0)]$ |

| Patch($[\bar{rsu}]$).Verified() $\rightarrow [\bar{rsu}] \cup area.SD$ |

end

for $iter \in ITERS$ do

for $rsu \in area.SD.Reverse()$ do

| $\bar{rsu} = area.SD.Random()$ |

| Swap($\bar{rsu}, rsu$) $\rightarrow area.SD$ |

end

for $i \in [1, H)$ & $rsu \in area.SD$ do

| $rsu.b = [B \ast (i - 1) \sim B \ast i]$ |

end

Valid($\{BP_{mc}, area.SD\} \rightarrow area.SD$)

end
by authorities, and area-based RSU information $A$ is input to main-chain servers and parsed to Areas structure. Particularly, sharding scheme can be triggered upon receiving updates of RSUs from shards. In procedures, servers as access points retrieve previous sharding information Areas from BC, which is compared with the inputs for each area, and processes the area.RSUs which differ from the existing sharding scheme. Following Eq. 7.5 and Algorithm 3, the popularity $p_{rsu}$ and security $s_{rsu}$ of RSU $rsu$ can be calculated, and those qualified RSUs can be added into the related area sharding assignment area.$SD$. If the qualified RSUs are less than $M$, more RSUs should be patched and added. Iteratively, from this $M$ nodes and based on DPoS, main-chain shuffles their order and chooses $H$ as the BPs for each shard, and assigns $B$ blocks for each shard-BP. Eventually, the sharding scheme is sent to $\{BP_{mc}\}$ for validation and consensus. As using PBFT where $f$ honest, $f$ malicious, $f$ uncertain, and 1 primary nodes are assumed there, simply not only the number of nodes but also groups determine the validity of a block, such that block confirmation requires $2f + 1$ agreements where $f = \frac{1}{3} \times H$. The time complexity of PBFT three stages is estimated as:

$$T_{pbft} \approx O(3f) + O(3f \times 2f) + O((3f + 1) \times (2f + 1))$$

$$\approx O(H^2) \quad (7.6)$$

where in the first stage current producer sends a block to $3f$ validators, subsequently each validator makes decision and waits for at least $2f$ same responses to proceed, and finally in case the producer is untrusted, $2f + 1$ agreements are needed for each node to confirm and preserve the block.
7.3 System Evaluation

The evaluation of system targets the accuracy of security indexing, the latency and throughput in sharing data of transactions and blocks. Continuous generation and transmission of transactions is automated to perform stress test in our system. In terms of training for security indexing, we use Cytoscape [104] with Range-Network apps to simulate the system topology and device attributes. We develop the entire system by improving the vDLT which is a permissioned BC platform [141]. To simulate the interaction between VANET and BC, we add modules into NS3 [96] which can simulate the VANET communication and send the data packets to our system devices. The benchmark

7.3.1 Experiment Configuration

Majorly, the proposed system is constructed in a cluster of virtual machines, which are generated by installing VMware vSphere in a set of servers and computers. Table 7.1 lists the physical and virtualized devices, their major resource configuration and running systems. According to Table 7.1, the dataset generation and training is performed in a separate computer which runs our BC client system to with the virtualized environment. There are 10 virtual machines (VMs) used as BS which form the main-chain and about 50 VMs as RSUs to form 3 shard-chains. The whole resource is available for running up to 200 light VMs as the BC clients.

The are two sets of virtual machines created which respectively are installed with our proposed system and original vDLT to facilitate the switch between two architectures. In terms of BC supports, the relevant SCs are developed and pushed into the proposed system, which are implemented based on the mechanism of SCs in vDLT. In addition, communication in virtualized environment is supported with 1Gbps Ethernet ports. Virtual switches are configured to create subnets, and main-
chain and shard-chains are built in accordance with subnets. Particularly, one-hop communication is supported between vehicle and RSU virtual machines in order to simplify the routing procedures, and their links can be dynamically changed by being connected to different virtual switches ports so as to simulate the mobility of vehicles.

<table>
<thead>
<tr>
<th>Item</th>
<th>Num.</th>
<th>Cat.</th>
<th>Specs</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>7</td>
<td>Phy.</td>
<td>CPU (12-cores), RAM (192-GB)</td>
<td>vSphere</td>
</tr>
<tr>
<td>Computer</td>
<td>8</td>
<td>Phy.</td>
<td>CPU (8-cores), RAM (64-GB)</td>
<td>vSphere</td>
</tr>
<tr>
<td>AU. Node</td>
<td>1</td>
<td>Phy.</td>
<td>CPU (8-cores), RAM (64-GB)</td>
<td>BC-Client, Training</td>
</tr>
<tr>
<td>BS Node</td>
<td>10</td>
<td>Virt.</td>
<td>CPU (2-core), RAM (16-GB)</td>
<td>BC-Service</td>
</tr>
<tr>
<td>RSU Node</td>
<td>50</td>
<td>Virt.</td>
<td>CPU (1-core), RAM (8-GB)</td>
<td>BC-Service</td>
</tr>
<tr>
<td>Vehicle Node</td>
<td>200</td>
<td>Virt.</td>
<td>CPU (1-core), RAM (4-GB)</td>
<td>BC-Client</td>
</tr>
</tbody>
</table>

Table 7.1: The list of devices and related configuration.

7.3.2 Dataset and Training

The training of deep autoencoder generates model for security indexing which requires the dataset of devices features. The dataset used for training is generated via Cytoscape which can be used to configure complex networks and device features. In this project, Cytoscape is installed two extra apps, which are Range Network and OpenC2. Range Network supports the configuration of cybersecurity attributes in devices, with OpenC2 to provide automation of network creation. The dataset in our training covers the features of **Vulnerability**, **Service-software**, **Service-port**, **Client-application**, **System-process**, **Firewall-rule**, **System-account**, **Network-interface**, **Data-volume**, **Device-reachability**, **Data-rate** and **Packet-size**. As proposed, these features of devices are arranged in images and assigned different colors. The number
of features in devices determines the size of input images to training.

The evaluation of training deep autoencoder targets the loss. The neural networks on encoding and decoding sides have symmetric structure. There are 4 fully-connected layers constructed for deep learning, and according to the size of input images the number of encoded outputs differ. Fig. 7.6 compares the training loss between two structures. The losses pertain to inputs of 32 × 32 images which are compressed into 4 × 4 size file, and inputs of 48 × 48 images encoded to 8 × 8 size. According to Fig. 7.6, less than 5 epochs of training can generate a feasible model, and the results are consistent between training dataset and validation dataset. In addition, while the encoded output size 8 × 8 is bigger than 4 × 4, the smaller loss is found between decoded output and original input. It is inefficient to increase hidden layers, since small losses and few epochs are witnessed in process, and more layers accordingly increase the resource consumption which is impractical in VANETs.

Subsequently, with the model of encoding images, we perform clustering over the encoded outputs to find the suitable centroids for device security rankings. Fig. 7.7
Figure 7.7: The distances in K-Means clusters.

shows the result of clustering via K-Means. The related evaluation focuses on the similarity of data inside each cluster, such that in each cluster if the distance between data points and the centroid is shorter, the accuracy of clustering is higher. The calculation of distance follows the MSE concept which compares the pixel vector between encoded data and the centroid. In Fig. 7.7, it is shown that as more clusters are designed, the distances in each cluster are smaller, and meanwhile, the encoded rate influences the clustering performance, since more features extracted can help improve comparison reliability but will require more computing resource.

7.3.3 Security Indexing

As using ZTA, access control is dependent of security assessment, and our security indexing approach is performed to locate the closest cluster centroid with the compress image of device features. For trusted storage, the trained model and cluster assignment are preserved in our BC system, and they are retrieved by querying the BC ledger for
assessing device security status. Therefore, security indexing gains considerable protection. Moreover, the performance metrics of latency and throughput are imperative in evaluating the security indexing, since they can reflect the efficiency and reliability. Security indexing mostly is triggered by opening account when feature uploads are required, therefore we focus on sending open-account transactions to evaluate security indexing.

The latency of security indexing incorporates the time in sending feature images, encoding the images, finding the closest centroid, and returning the indexing results. Fig. 7.8 shows the latency comparison between two kinds of images. As in encoding process, the number of clusters $K$ is much smaller than the size of input, the time complexity is determined by the sizes of original input and encoded output. According to Fig. 7.8, the latency in processing $32 \times 32$ images is lower than in $48 \times 48$, but the gap is mostly about 50 ms. The beginning larger difference is caused by the burst of transactions, while the volume becomes steady, the latency is reliable.

The throughput during security indexing can reflect the reliability of process via
transaction sharing. This metrics is related to the sizes of images, and in our evaluation, the larger inputs are encoded in lower compression ratio in order to reduce the loss of information. Thus, throughput is also compared between $32 \times 32$ and $48 \times 48$ images. Fig. 7.9 shows the throughput differs less than 1Mb. In comparison, the bigger image can store 1KB more than the smaller one, which causes the difference. Importantly, it is described in Fig. 7.9 that the throughput trends are smooth except the beginning increases, but their difference keeps in a steady level even is faced with stress testing, which means that the data is transmitted reliably.

7.3.4 Transaction Sharing

The delay and success rate of sharing transactions are the important performance metrics to evaluate our system. The entire of sharing a transaction starts from getting SCs, then building and transmitting the transaction, and finally ends by wrapping the transaction in blocks. Since sharding is designed into our framework, it is necessary to compare the cross-shards and single-chain sharing solutions. Besides, the ideal
situation is to send their transaction to the shard where sending devices or users open accounts, such that the transaction is swiftly shared in a small area.

The latency comparison is shown in Fig. 7.10 where three cases are considered. As in Fig. 7.10, Inner-Shard and Cross-Shards cases run our proposed hdPBAC which leads to certain latency, and Single-Chain in which only one chain is built gives the most complicated situation, whereas the Inner-Shard which means the ideal situation shows the most clean and smooth trend of latency variation. In Single-Chain, the communication environment and the device being BP highly affect the latency. At the beginning, as transactions being sent, the first one causes the longest latency, and the last one using the shortest time. With only one chain, in the time period of block production, it is difficult to guarantee successful procedure. The lowest latency of transaction sharing implies the re-transmission of lost transactions, such that the latency can tell the time confronting stress testing. The Cross-Shards case includes Inner-Shard and Inter-Shard communication, and main-chain coordination is involved, thus its complexity is similar to Single-Chain. However, the Inner-Shard procedures
digest part of complicated issues. Inner-Shard is ideal, and it is possible if the shard is big to cover a large, but if the area is too big, the sharding scheme falls back to the Single-Chain case. In our evaluation, there are 3 shards, and the transaction amount is about 1000, therefore, the uniform distribution of these transactions into shards makes transaction sharing reliable and efficient.

In terms of success rate of transaction sharing, we consider different sending rates as the stress test level, and hence analyze the packet loss to find the system robustness. Fig. 7.11 illustrates the success amount in terms of various sending rates. In comparison, it is found that in Fig. 7.11 the Single-Chain design hardly cope with the stress test. The reason is that current BP is overloaded which needs to receive the amount of transactions, execute the related SCs to validate the transaction, processing the blocks in the ledger. Even though hdPBAC requires extra workload, the 3 shards can relieve the burdensome, and our sharding scheme chooses the idle RSUs to increase the computing capability. In particular, the three cases use the same devices and
configuration and their evaluation is reliable.

7.4 Remarks

The utilization of BC technology for protecting VANETs has been widely studied, and while ZTA gains increasing attention on improving cybersecurity, the integration of ZTA, BC and VANET is believed promising to forge the innovative protection solution. VANETs are accompanied with the issues majorly caused limited resource and infrastructure-less topology, and hence the traditional perimeter-based security solution which requires complex hardware and software configuration becomes impractical in the ad-hoc communication environment. BC technology is effective for trust and privacy protection, but its decentralization mechanism has conflicts with the capability of VANETs. Scaling BC networks can balance the reliability and performance requirements, together with ZTA constructed for VANETs, it can effectively solve the BC trilemma issues. In this paper, we propose a framework considering the highly synthesis of BC and ZTA into protecting the data sharing in VANETs. The major components of framework are designated and their relations are explained. We develop the framework to demonstrate the significance of our proposed framework. Dataset features are designed and generated by simulating complex networks; considering the VANET realistic, a lightweight BC-related security assessment approach is developed and sharding scheme according to traffic and security situation in VANETs is designed; an access control model is proposed to support ZTA. This paper focuses on providing a practical solution for VANETs using BC and ZTA technologies to achieve enhanced security solution.
Chapter 8: Conclusion and Future Works

8.1 Conclusion

The design of vDLT-based DDL is proposed into the VANETs to improve the object detection for protecting the safety of road traffic. The proposal investigate the important domains which require improvement for this eventual objective, and accordingly divide the work into multiple stages. The designs in stages are related with each other, and the outcomes from previous stage serve as the supports for the next stage which considers improvement for higher-level requirement. Stage one contributes a practical system for real-time surveillance via vDLT and a framework for constructing a vDLT-based VANET; and the contribution builds up the supports for multimedia transmission in vDLT platform and subsequently the distribution over VANETs via virtualization technology. Stage two produces a system which protects the transmission of traffic events for vDLT-based VANETs; the features of layered-design inside vDLT provides better reliability and scalability of multimedia networking for vDLT-based VANETs. Stage three focuses on the DDL with the supports of transmitting big-volume data and managing the data trust in vDLT which are generated from the first and second stages; vDLT-based DDL demonstrates higher reliability in comparison with CC-based distributed training. Stage four proposes a framework which integrates ZTA and sharding supports into BC-based VANETs to further protect the data sharing, where ZTA access control takes device security index as the primary attribute along with control policies adapted to network variation, and sharding schemes exploit security indexing and traffic statistic together with SCs as interfaces to swiftly execute ZTA policies and cross-shards management.
8.2 Future Works

In developing the proposed system, there are some challenges countered, which mostly pertain to the deployment and evaluation of system. Due to the distribution requirement, there numerous devices need to be managed, and especially when functions or components are updated, manual deployment of system into devices becomes inefficient and complicated. For practical consideration and stable result reproduction, some baselines and dataset chosen have been well implemented; but for further improvements and academic purpose, recent or even experimental supports can be added to system evaluation, and the features added should be investigated in terms their impacts on system feasibility.

Besides, the research on DDL and distributed OD in VANETs using BC and ZTA can be improved by considering the compression of model and the caching of training dataset, at the major goal to further improve the performance of proposed system. It is well accepted that the communication condition is limited to the increasing amount of parameters shared across the distributed training workers of DDL, and thus it is promising to compress the model for reduced data size in DDL process. Meanwhile, as the schemes of feeding dataset for training into DDL workers impact the DDL performance, dataset distribution should consider the computation capability and the communication environment of each worker. As the final accomplishment of training is determined by the slowest work, it is significant to optimize the training task assignment and the dataset provision fashion to coordinate the training schedules among all the DDL workers.

Last but not least, it is worth continuing on the improvement AI and BC technologies, as AI and BC technologies are seriously influencing the real world. It is imperative to keep continuous effort on in-depth and extensive studies to create advanced designs so as to empower the AI and BC significance.
List of References


