Delay-Aware Hash Tree for Blockchain on Vehicular Delay Tolerant Networks

Joong-Lyul Lee[1(✉)] and Joobum Kim[2]

[1] Department of Mathematics and Computer Science, University North Carolina at Pembroke, Pembroke, NC 28372, USA
joonglyul.lee@uncp.edu

[2] Department of Information Technology, Middle Georgia State University, Macon, GA 31206, USA
joobum.kim@mga.edu

Abstract. Blockchain (BC) technology is being applied to various applications with a lot of interest in data security. One of these applications is being applied in Vehicular Ad-hoc Networks (VANETs). Due to the high mobility of VANETs, frequent changes of the network topology occur and the network connection can be lost. These cause a high delay and network partitioning. Therefore, a new research paradigm called Vehicular Delay Tolerant Networks (VDTNs) is introduced to study these issues. When BC technology is applied to VANETs, it constructs a data gathering tree or schedules a routing path in the network layer for efficient data collection and handling network disruption. Furthermore, frequent topology updates happen. However, if the BC hash tree is not updated in the application layer, this causes an increase in network delay due to the inconsistency between the data gathering tree and the hash tree. In this paper, we describe this problem and propose new greedy algorithms to construct a hash tree that considers the network connection time on VDTNs. In such a hash tree, the inconsistency between the hash tree and the data gathering tree can be resolved by constructing the data gathering tree in consideration of the network connection time. Additionally we analyze the time complexity of the proposed algorithms.

Keywords: Blockchain · VANETs · VDTN · Hash tree

1 Introduction

Blockchain (BC) technology is receiving a lot of attention along with the popularity of data security and is a very stable, secure technology due to the characteristics of distributed technology. As an example of data security in BC, many mobile devices record online shopping and payment history. These histories can be altered or forged by cyber attackers. If this information is created using BC, it can be protected against data falsification from cyber attackers. The representative application technology using the BC is the cryptocurrency that is currently attracting much attention [1].

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2023
Published by Springer Nature Switzerland AG 2023. All Rights Reserved
Z. J. Haas et al. (Eds.): WiCON 2022, LNICST 464, pp. 91–101, 2023.
https://doi.org/10.1007/978-3-031-27041-3_7

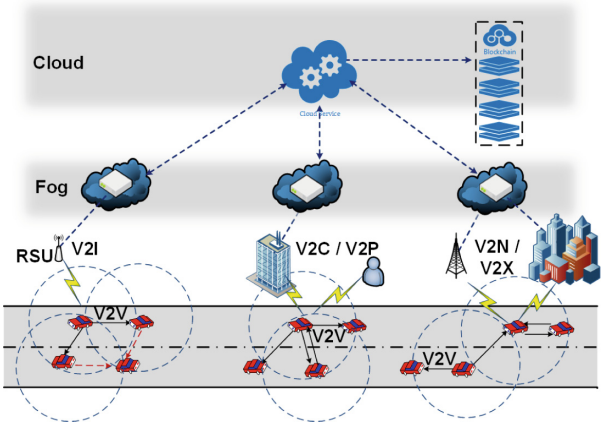

Fig. 1. A Vehicular-Fog-Cloud (VFC) Framework for BC

This BC is being applied to Vehicular Ad-hoc Networks (VANETs) [2,3]. Figure 1 presents a Vehicular-Fog-Cloud (VFC) framework when BC is applied to VANETs. VANETs are connected to five different types of communication devices. These five different types of communication devices are as follows: Vehicle to Infrastructure (V2I), Vehicular to Vehicle (V2V), Vehicle to Cloud (V2C), Vehicle to Pedestrian (V2P), Vehicle to Network (V2N), Vehicle to Everything (V2X) [4,5]. VANETs have high network delay, frequent network disconnection, and partitioning due to high mobility. A new research paradigm called vehicular delay tolerant networks (VDTNs) is introduced to study these issues. Delay-tolerant networking (DTN) continues to provide network connectivity even when heterogeneous network failures occur. VDTNs are similar to the characteristics of DTN in that network partitions or network disconnections occur [6,7].

When BC technology is applied to VANETs, it does not consider highly dynamic network topology [8,9]. VANET nodes construct a network topology for a short period to transmit data to the Cloud data center through Fog nodes as shown in Fig. 1. Due to the movement of VANET nodes over time, the network topology for transmitting data changes, and then the hash tree (eg. Merkle tree [10,11]) in BC does not change. This discrepancy between the data gathering tree and the hash tree in BC causes the network delay to increase. To compensate for this shortcoming, we propose new greedy algorithms to construct a hash tree for BC based on a data scheduling graph.

Contribution of this paper: The main contributions of this paper can be summarized as follows:

1. When the BC is applied to VANETs, the network topology changes frequently due to the high mobility of the VANET. We notice that the discrepancy between the changed network topology and the hash tree in the BC causes a problem of increasing network delay, and define this problem in this paper.

2. We propose new greedy algorithms that construct a hash tree for BC based on the data collection schedule for VANETs with high mobility and describe the time complexity of the proposed algorithms.

To the best of our knowledge, this is the first research work about network delay caused by the discrepancy between the hash tree and data gathering tree (routing path) in BC technology of VANETs, where the network topology changes most frequently over time.

The rest of this paper is structured as follows. Section 2 describes the network model, problem formulation, the proposed algorithm, and the analyzed result. Finally, Sect. 3 concludes this paper.

2 System Model and Design

In this section, we describe the core components in the system, notation, problem statement, and the proposed delay-aware hash tree construction algorithms.

2.1 Core Components in the System

Vehicle Node (VN). A vehicle with a wireless onboard unit (OBU) is connected to an Internet through a Road Side Unit (RSU) which is connected to the backbone network around the road. Each VANET node transmits necessary safety-related information for driving such as road surface condition information, traffic information, weather information, etc. to the Cloud center through the Fog node by communicating with other nodes or RSU.

Cluster Head Node (CHN). The VANET node constructs a topology to collect safety-related information or provide network connectivity to each node. This node that constructs and controls the topology is the cluster head node (CHN).

Road Side Unit (RSU). The Road Side Unit (RSU) connected to the Backbone network is installed on the roadside and provides network connectivity to passing cars, thereby providing Internet connectivity.

Fog Node (FN). The information collected by the VANET nodes is sent to the Cloud center, and the information from numerous VANET nodes is concentrated in on the data center, creating a network bottleneck. To overcome these shortcomings, nearby servers (Fog nodes) provide connectivity and accessibility. These Fog nodes (FN) process the information collected from the VANET nodes and send them to the Cloud center.

Cloud Data Center (CDC). The information collected from the Fog nodes is stored in the Cloud data center (CDC), and the stored information is analyzed. The analyzed useful information is provided to each client.

2.2 Notation and Problem Statement

Definition 1 (Graph by vehicular node v). *Given a graph $G = (V, E)$ by each VANET node v_n and each edge $E = (v_i, v_j)$ existing within the transmission range of VANET node v_i and v_j.*

Definition 2 (Communication range of vehicular node v). *The communication range c_{v_i} of a node is defined as the communication range of each VANET node v_i.*

Definition 3 (Network connection of vehicular node v_i and v_j). *The network connection $N(v_i, v_j)$ of VANET nodes v_i, v_j is defined as the minimum communication range of VANET nodes v_i, v_j, i.e. $N(v_i, v_j) = min\ (c_{v_i}, c_{v_j}) : v_i, v_j \in V\}$.*

Definition 4 (Graph by time space). *Given a graph $G = (V_{(n,t)}, E_{(i,j,t)})$ by time spaces and each VANET node has location $V_{(n,t)} = (x_t, y_t)$ and each edge $E_{(i,j,t)} = (V_{(i,t)}, V_{(j,t)})$ in the time space.*

Definition 5 *Network Connection* metric space). *Consider the Network Connection metrics space $(V_{(i,j)}, \delta_{(i,j)})$, where v_i and v_j are in the set of VANET nodes. For each pair of VANET nodes $v_i, v_j \in V$, $\delta_{(i,j)}(StartTime\_stamp(st), End\ Time\_stamp(et))$ is an estimated network connection time duration.*

1. $\delta_{(i,j)}(st, et) >= 0$
2. $\delta_{(i,j)}(st, et) = \delta_{(j,i)}(st, et)$ if $v_i \neq v_j$
3. $\delta_{(i,j)}(st, et) = 0$ if $v_i = v_j$

Problem 1. *Given a set of the vehicular set $V = \{v_1, v_2, v_3, \cdots, v_n\}$ and time constant $t \geq t_{max}$, we want to construct a hash tree with no additional network delay based on a data gathering tree to collect data efficiently on predictable VDTNs which provide continuous network connectivity despite frequent network failures.*

As shown in Fig. 2-(a), VANET nodes construct a data transmission tree to send and receive data efficiently. However, this data transmission tree is changed continually as time goes by due to the mobility of VANETs. If the BC Hash tree and VANET data transmission tree are the same as in Fig. 2-(a), there will be no delay beyond the expected network delay. If the VANET data transmission tree is updated, there may be a case of inconsistency with the BC Hash tree.

As shown in Fig. 2-(b), when the data transmission tree is updated within the same tree, the data is transmitted inefficiently. For example, when the cluster head (Black node) changes to the terminal node due to the VANET node's

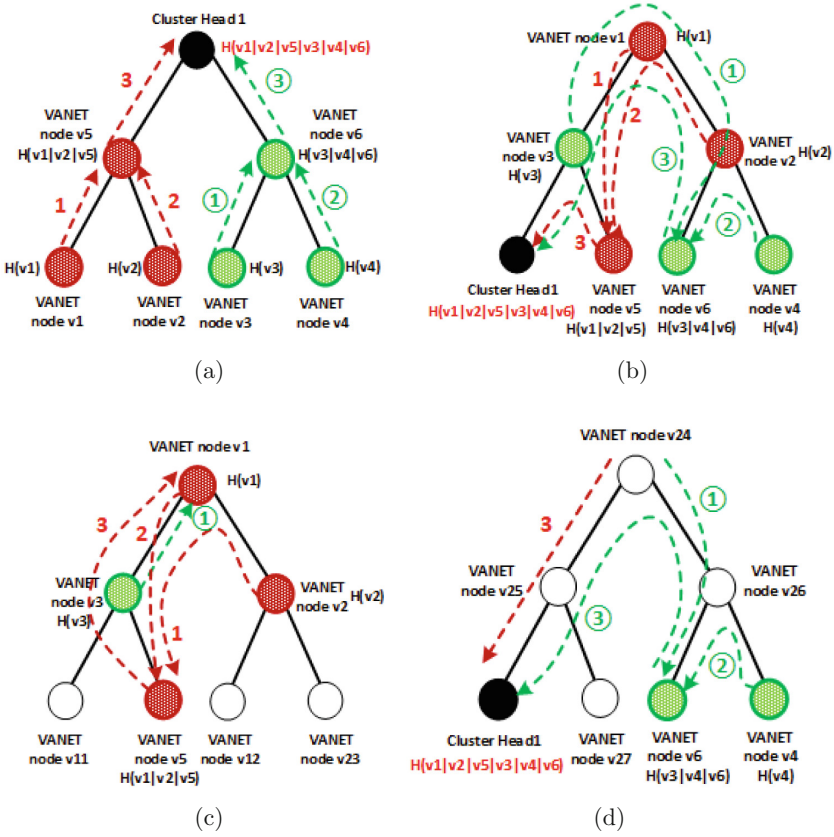

Fig. 2. Discrepancy between data gathering tree and hash tree on VDTN: (a) A case where the data gathering tree and hash tree are identical, and (b) A case where the data gathering tree and hash tree are not the same, (c), and (d) A case where the nodes of the hash tree are split into two data gathering trees.

mobility, VANET node 1 and VANET node 2 are forwarded to VANET node 3 after hashing the data with the previous terminal nodes(Red arrows 1, 2, and 3). After calculating the Hash value added with its data value, VANET node 3 is transmitted to the current end node, which is the head of the previous network transmission tree. VANET node 3 and VANET node 4 transmit their data hash values to VANET node 6. Then, after calculating the hash value of the received data and the hash value of its data, the VANET node 6 transmits the hash data to the cluster head which is changed to the current terminal node(Green arrows 1, 2, and 3).

Figures 2-(c) and (d) show the case in which the nodes in Fig. 2-(a) are mixed with other nodes to form two new data gathering trees due to the mobility of VANET nodes. As the cluster head (Black node) exists in the tree in Fig. 2-(d), VANET node 1 and VANET node 2 in Fig. 2-(c) transmit the hash value to VANET node 3. Then, this hash value is again transmitted to the cluster head (Black node) in the terminal node in 2-(d) (Red arrows 1, 2, and 3). VANET node 3 in Fig. 2-(c) and VANET node 4 in Fig. 2-(d) transmit the hash value to VANET node 6, and VANET node 6 transmits the hash value to the cluster head (Green arrows 1, 2, and 3). As shown in this example, the data collection tree is frequently updated based on the network layer. However, a Merkle tree is used to calculate the data hash value in the BC, and this tree does not update when the data gathering tree is changed.

In a DTN, many existing algorithms consider scheduling to transfer data, but there is no algorithm considers both scheduling and delay. As seen in the example in Fig. 2, delay as well as scheduling for data delivery are important parts that cannot be ignored.

Therefore, we propose new algorithms considering scheduling and delay for BC on VDTNs.

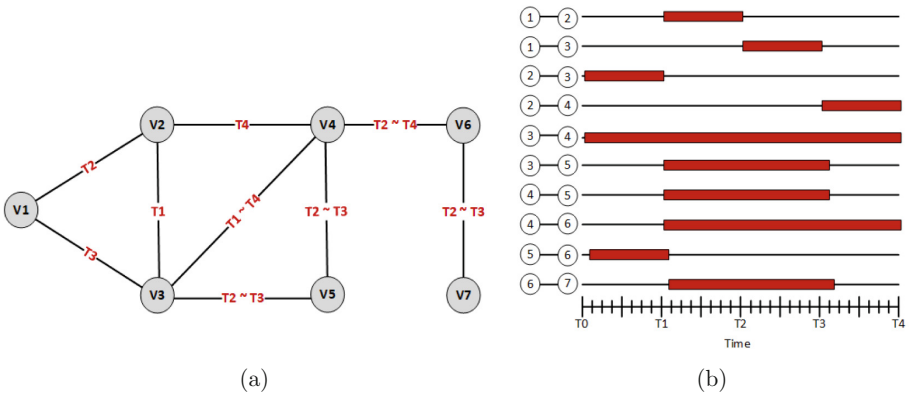

Fig. 3. A complete graph of the time-evolving network

As shown in Fig. 3-(a), it creates a complete graph based on a network connection and time-evolving network. In a time-evolving network, each VANET node has a network connection and network duration with the neighbor node in different time slots as in Fig. 3-(b).

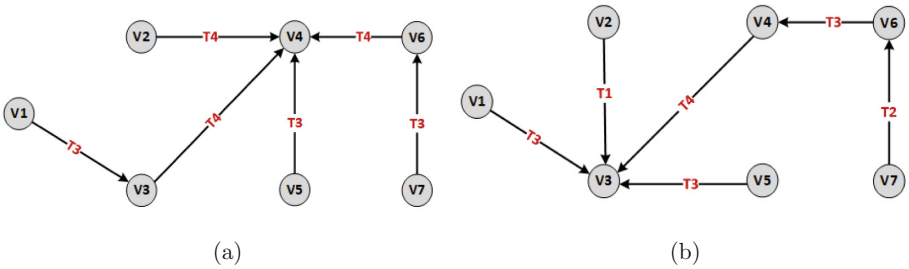

Fig. 4. Scheduling graphs for data collection based on Fig. 3

2.3 Delay-aware Hash Tree on Predictable VDTNs

Phase I: VANET Set-Up. As the movement of the VANET nodes is predictable, it creates a complete graph based on network connected time. This research method has been studied in many papers [12,13]. Furthermore, it has a connected schedule between nodes during a specific time as in Fig. 3-(b). It selects a root node set choosing the highest time-stamp with the longest network connection time. In Fig. 3-(b), the time-stamp with the longest connection time is the connection link of nodes 3 and 4. Therefore, nodes 3 and 4 are included in the root node candidate sets.

Phase II: Scheduling Data Collection. Node 3 and node 4 are the root node candidates. When node 4 is selected as the root node, node 4 selects the highest timeslot in the connecting link with the neighboring nodes as shown in Fig. 4-(a). Node 4 selects nodes 2, 3, 5, and 6 as neighboring nodes, and selects the highest timestamp that can also select timestamps. If there is a node that has not been selected in this round, the node with the highest time-stamp among the neighboring nodes connects with the neighboring node. However, the selected neighbor nodes choose a timestamp that is lower than the timestamp selected by the root node but is selectable higher. In Fig. 4-(a), node 3 selects node 1 with timestamp 3, and node 6 selects node 7 with timestamp 3 in the next round. In Fig. 4-(b), node 3 is selected as the root node. Neighbors 1, 2, 4, and 5 are selected by selecting the highest timestamp that node 3 can select among the neighboring nodes. Since there is a node that was not selected in this round, node 4 among the child nodes selects node 6 in the next round. As there are still unselected nodes, node 6 checks circularity and selects child node 7 in the next round.

Algorithm 1. Data Scheduling algorithm based on VDTN

Input: a set of vehicle V^*, a set of time stamps and duration T^*;
Output: a cluster set C^*;

1:
2: $S \leftarrow$ a set of all visiting nodes;
3: $V \leftarrow$ a set of all vehicle nodes;
4: $T^*_{List} = Sort\{\delta_{(i,j)}(st_1, et_1), \delta_{(k,l)}(st_2, et_2)...\}$ by endTimeSlot;
5: R^* = select the root node candidate group among the nodes with the longest and highest timestamp (et) from T^*_{List};
6:
7: **while** $V \neq \{\emptyset\}$ **do**
8: Pick a node v_i from R^* and remove node v_i from V;
9: Add a node v_i to S and C^*;
10: Select the neighbor node of node v_i and add neighbor node v_j to S;
11: Select the highest time slot of node v_j;
12: **if** *timestamp of* $v_j \leq$ *timestamp of the root* **then**
13: Add child node list C^*;
14: remove node v_j from V;
15: **end if**
16: **end while**
17:
18: Return cluster set C^*;

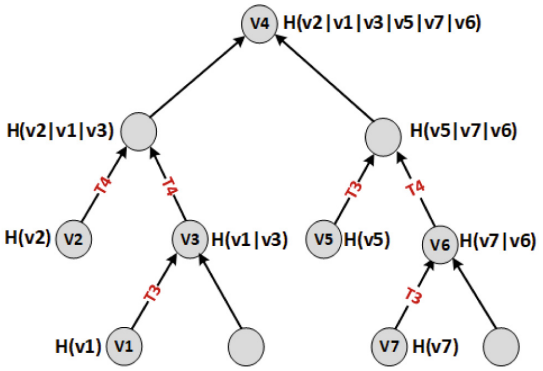

Fig. 5. A Dealy-aware Hash Tree based on Fig. 4-(a)

Algorithm 2. DHT construction algorithm

Input: a set of cluster C^*;
Output: hash tree H;

Pick a root node v_i from C^* and remove the node from C^*;
$H\_root = v_i$;
for i to n **do**
    $child\_node$ = $child\_node$ of v_i;
    if $child\_node.size > 0$ **then**
        AddChildNode($child\_node$);
    end if
    $i++$;
    $H_{level}++$;
end for

function AddChildNode($child\_node$)
    if $child\_node.size == 1$ **then**
        Add $H.left\_child \leftarrow child\_node$;
        Add $H.right\_child \leftarrow dummy\_node$;
        $child\_node.size$=$child\_node.size - 1$;
    else if $child\_node.size == 2$ **then**
        Add $H.left\_child \leftarrow child\_node$;
        Add $H.right\_child \leftarrow child\_node$;
        $child\_node.size$=$child\_node.size - 2$;
    else if $child\_node.size > 2$ **then**
        Add $H.left\_child \leftarrow dummy\_node$;
        Add $H.right\_child \leftarrow dummy\_node$;
        $left\_child\_node.size$=$\lceil child\_node.size/2 \rceil$;
        $right\_child\_node.size$=$\lfloor child\_node.size/2 \rfloor$;
        AddChildNode($left\_child\_node$);
        AddChildNode($right\_child\_node$);
    end if
end function

Return hash tree H;

Phase III: Delay-aware Hash Tree (DHT). The root hash value of the hash tree stored in BC. This hash tree is a binary tree structure. The data gathering schedule trees in Fig. 4-(a) and (b) are not binary trees. Therefore, after constructing a tree considering scheduling for data gathering, this tree is converted into a binary tree as in Fig. 5. It pops up the root node from the cluster set that is the output from Algorithm 1. As shown in Fig. 5, when there are two or more child nodes, the level of the tree is increased and a dummy node is added. The algorithm terminates when there are no more child nodes.

A hash value $h_{(j,k)} = H(v_i)$ is computed with $1 \leq k \leq 2^n$. The leaf node of the hash tree has level $j = 0$ and the root node has level $j = n$. The number of

nodes in a level is from left to right. Therefore, $h_{(0,0)} = H(v_1)$, $h_{(1,0)} = H(v_1|v_2)$ and $h_{(2,0)} = H(v_1|v_2|v_3)$.

Theorem 1. *The time complexity of the proposed Algorithm 1 on VDTN is O(E).*

Proof. Initially, an algorithm selects a root node in a cluster. It executes as many as the number of VANET edges. Therefore, the time complexity of the proposed algorithm is $O(E)$.

Theorem 2. *The time complexity of the proposed Algorithm 2 is $O(log_2 V)$.*

Proof. This algorithm is to convert from a tree set for efficient data collection to a binary tree. And it repeats as many as the level of the binary tree. Therefore, the time complexity of the proposed algorithm is $O(log_2 V)$.

3 Conclusion

A data gathering tree constructs for efficient data collection in VANETs. It frequently changes due to high mobility. However, the hash tree in BC is not updated like the data gathering tree when BC is applied to VANETs. If the data collection tree and hash tree are not identical, then there is a problem in which network delay increase. In this paper, we describe this problem and propose algorithms for constructing a tree that collects data at a specific time based on the mobility expected as a solution and construct a hash tree based on this data gathering tree on VDTNs. We showed the time complexity of the proposed algorithms. In future work, we have a plan to simulate these algorithms and perform performance comparisons with the other algorithms.

References

1. Nakamoto, S..: Bitcoin: A peer-to-peer electronic cash system
2. Li, H., Pei, L., Liao, D., Sun, G., Xu, D.: Blockchain meets VANET: an architecture for identity and location privacy protection in VANET. Peer-to-Peer Netw. Appl. **12**(5), 1178–1193 (2019)
3. Deng, X., Gao, T.: Electronic payment schemes based on blockchain in VANETs. IEEE Access **8**, 38296–38303 (2020)
4. Lee, M., Atkison, T.: VANET applications: past, present, and future. Veh. Commun. **28**, 100310 (2021)
5. Cunha, F., et al.: Data communication in VANETs: protocols, applications and challenges. Ad Hoc Netw. **44**, 90–103 (2016)
6. Kang, H., Ahmed, S.H., Kim, D., Chung, Y.-S.: Routing protocols for vehicular delay tolerant networks: a survey. Int. J. Distrib. Sens. Netw. **11**(3), 325027 (2015)
7. Cheng, P.-C., Lee, K.C., Gerla, M., Härri, J.: GeoDTN+ Nav: geographic DTN routing with navigator prediction for urban vehicular environments. Mob. Netw. Appl. **15**(1), 61–82 (2010)
8. Kim, S.: Impacts of mobility on performance of blockchain in VANET. IEEE Access **7**, 68646–68655 (2019)

9. Xie, L., Ding, Y., Yang, H., Wang, X.: Blockchain-based secure and trustworthy internet of things in SDN-enabled 5G-VANETs. IEEE Access **7**, 56656–56666 (2019)
10. Merkle, R.C.: Secrecy, Authentication, and Public Key systems. Stanford university (1979)
11. Becker, G.: Merkle signature schemes, merkle trees and their cryptanalysis. Ruhr-University Bochum. Technical report (2008)
12. Lee, J.-L., Hwang, J., Park, H.-S., Kim, D..: On latency-aware tree topology construction for emergency responding VANET applications. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 57–63 IEEE (2018)
13. Lee, J.-L., Hwang, J., Park, H., Kim, D.: Ad hoc communication topology construction for delay-sensitive internet of vehicle applications. Wirel. Commun. Mob. Comput. 2022 (2022)