Poster: Time Analysis of the Feasibility of Vehicular Blocktrees

Joshua Joy
University of California - Los Angeles
jjoy@cs.ucla.edu

Greg Cusack
University of California - Los Angeles
gregcusack@ucla.edu

Mario Gerla
University of California - Los Angeles
gerla@cs.ucla.edu

ABSTRACT

In this paper we evaluate the feasibility of the vehicular blocktree proposed by Joy [4] whereby vehicles write and aggregate their signatures to a blockchain. We analyze the end to end cycle time to collect, write, and persist the content to the blockchain as well as the total minting time. We show via time analysis that the cycle time occurs on the order of hundreds of milliseconds, showing that the proposed design is indeed feasible.

1 INTRODUCTION

In this paper we evaluate the feasibility of the vehicular blockchain proposed by Joy [4] whereby vehicles write and aggregate their signatures to a blockchain. We analyze the end to end cycle time to collect, write, and persist the content to the blockchain as well as the total minting time. We show via time analysis that the cycle time occurs on the order of hundreds of milliseconds, showing that the proposed design is indeed feasible.

Content is addressable by the signature pointers and is routed by content based routing (e.g., information centric networks or delay tolerant networks). The pointers ensure that the data is tamper proof and also enables querying for the data (e.g., tags or attributes). The consensus protocol ensures that the content pointed to by the signatures is validated and verified. Thus, false or fake data does not become stored.

2 EVALUATION

2.1 Simulation Model Setup

We quantitatively analyze the various elements of the network. The simulation model is similar to the model used in [6]. Our model is setup with a six lane highway, three lanes in each direction. We assume an inter-vehicle distance of 30m and a constant velocity of 100kmh. Furthermore, the roads are divided up into 120m segments due to the maximum working range of the widely used Velodyne LiDAR [5]. This means that each vehicle can communicate and monitor four vehicles per lane over six lanes. Vehicles in each 120m segment, receive signatures from all twenty-three other vehicles in the segment. Each 120m segment is associated with its own sidetree. The vehicles that mints in a segment writes blocks to the segment's sidetree.

2.2 Cycle Time Overview

In order for our system to be practical and safe, the entire block minting cycle time needs to fall close to the message transmission cycle time outlined in [6]. Every vehicle needs to do the following:

- Collect sensor data
- Generate keys
- Sign data
- Transmit data to edge cloud
- Broadcast signatures to surrounding vehicles
- Verify incoming signatures
- Access edge cloud
- Verify surrounding vehicle data with own sensor data
- Aggregate signatures into a block

Once a vehicle is (randomly) selected as the official aggregator, it will broadcast its block to the surrounding vehicles. The remaining vehicles need to compare the incoming block with their own and either accept or reject the block.

The rest of this section will walk through the time it takes each vehicle to perform the above actions.

2.3 Sensor Data Collection and Broadcast Time

Each vehicle needs to collect both its personal data such as velocity, acceleration, and location, but it also needs to identify the characteristics of its surroundings. The identification of other vehicles and their respective speeds and accelerations is done by LiDAR in combination with radar. The sensor data collection can all be done in parallel, but identification of neighboring vehicles takes the longest at a total time of 160ms [8]. Vehicle-to-vehicle signature transmitting delay is approximately 30ms [6].

2.4 Relevant Key Times

Key generation, signing, and verification times needed to be benchmarked using the ED25519 encryption standard using a commodity processor. SafeCurves [3] benchmarks using an Intel Westmere 2.4GHz, quad-core processor while using batches of 64 keys at a length of 64 bytes shows that key generation time took a mere 0.0025ms per key, while signing took a similar 0.0092ms. Verification required more time resulted in 0.014ms per key. However, with total overhead and latency, verification took 4ms [1].

2.5 Choosing and Accessing the Edge Cloud

We take advantage of the reliability and performance of a content delivery network (CDN) to provide vehicle large data needs. The requests are satisfied by the servers geographically closest to the location of the incoming request. In order to choose a proper CDN, we looked at latency and throughput performance of the network. Table 1 shows the performance metrics of the three main networks investigated. It can be seen in Table 1 that CacheFly, while hav-

SMARTOBJECTS'17, October 16, 2017, Snowbird, UT, USA.
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5141-6/17/10…$15.00
https://doi.org/10.1145/3127502.3127516

Table 1: Content Delivery Network Performance Metrics [7]

Metric	Akamai	CacheFly	EdgeCast
Latency (ms)	56.88	54.71	52.49
Throughput (Mbps)	2.07	3.88	3.37

ing a slightly higher latency than EdgeCast, provides the highest throughput. Looking into the future of a blocktree-based vehicular network, data sizes are likely to increase due to the increase in sensor resolution. As a result, in order to efficiently fulfill the need for larger data packets in the future, CacheFly was selected as the CDN for data delivery. The data provided in Table 1 are averages. Furthermore, latency and throughput times will vary over time as traffic varies and CDNs improve network performance. Therefore, it is likely that the network will require the use of multiple CDNs with varying performance characteristics. For network performance analysis and proof of concept evaluation, CacheFly is used in our current model.

It is important to note that the maintenance of the edge cloud needs to be supported. Cloud storage is not free. The edge cloud infrastructure will likely be supported through vehicle taxes. Whether the vehicle manufacturer or the vehicle owner is taxed, the cost to support the infrastructure will end up falling on the vehicle owner. Everyone pays taxes to support road infrastructure and maintenance, and paying for cloud infrastructure would be no different.

2.6 Data Comparison Time

Once all surrounding vehicle data is collected by each vehicle, a vehicle needs to compare its view of the world with that of the its surroundings. The time requirement here is minimal since vehicle locations, velocities, and accelerations need to be compared with just twenty-three other vehicles. However, for example, when Vehicle A receives data from Vehicle B via the edge cloud, Vehicle A needs to verify the vehicle locations as reported by Vehicle B. This operation requires an extra twenty-three comparisons per vehicle. However, after running a simulation to test a vehicle's required comparison time, the resulting maximum time requirement per vehicle is a mere 0.014ms[1].

2.7 Aggregation Time Requirements

Recall that aggregation requires three steps. First, vehicle signatures are aggregated and hashed along with the previous block ID. Second, a new block is created by grouping together the aggregated signatures, the hash output which serves as the new block ID, and a pointer to the previous block. Finally, once a vehicle is selected to be the official aggregator, the vehicle broadcasts the block to its neighboring vehicles. Each block contains twenty-four 100 byte packets[2]. Also included is the 32-byte block ID and routing data for broadcasting. Therefore, the size of each block is about 3kB. A SHA-256 bit hash takes 8.12 cycles per byte (upper bound) for 1.5KB packets using a 3.31GHz, quad-core, 2015 Intel Core i5-6600 processor [2]. 8.12 cycles per byte correspond to 24,360 cycles. At a clock speed of 3.31GHz, hashing requires 0.00736ms. By far the largest time requirement of block minting is the transmission delay from broadcasting the block into the network. Block delivery time is likely to take around 50-60ms due to large packet sizes [6]. We assume the minter requires just over 60ms to generate and broadcast a block.

2.8 Total Aggregation Cycle Time Analysis

After outlining the aggregation requirement time, we are now ready to calculate the total cycle time required to gather data, verify it, aggregate a block, and broadcast it to the surrounding vehicles and the edge cloud. The total time is found to be around 334.033ms via the following equations[3].

$$\begin{aligned}(gen\_key + LiDAR\_identification) + data\_sign+\\ (T_x\_data\_to\_cloud + broadcast\_sig) + verify\_sig+\\ access\_edgecloud\_data + verify\_data + hash\_sigs+\\ broadcast\_new\_block = total\_minting\_time\end{aligned} \tag{1}$$

We can substitute in the actual numerical values to get the total minting time.

$$\begin{aligned}(.0025ms + 160ms) + .0092ms + (55ms + 30ms)+\\ 4ms + 55ms + 0.014ms + 0.00736ms + 60ms\\ = 334.033ms\end{aligned} \tag{2}$$

With vehicles traveling at 100kph ≈ 28m/s, minting segments of 120m, and a minting cycle time of 334.033ms, each vehicle will be involved in 12-13 minting cycles per segment before traveling to another segment and minting blocks to a new sidetree.

While 334ms is longer than the 300ms message transmission time described in [6], a 10% deviation for vehicles flowing smoothly is not ideal but is accepted at the beginning stages of development. As development continues, it is likely the total cycle time will reduce do to increased CDN performance and further parallelization.

3 CONCLUSION

In this paper, we have analyzed the feasibility of vehicular blockchains. We show that the vehicle to vehicle communication, signature generation and verification, and content storage occurs on the order of hundreds of milliseconds which should be suitable for the vehicular blockchain proposed.

REFERENCES

[1] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2012. High-speed high-security signatures. *J. Cryptographic Engineering* 2, 2 (2012), 77–89. https://doi.org/10.1007/s13389-012-0027-1

[2] Daniel J. Bernstein and Tanja Lange. 2017. eBACS: ECRYPT Benchmarking of Cryptographic Systems. https://bench.cr.yp.to. (2017).

[3] Daniel J. Bernstein and Tanja Lange. 2017. SafeCurves: choosing safe curves for elliptic-curve cryptography. https://safecurves.cr.yp.to/. (2017).

[4] Joshua Joy. 2017. Vehicular Blocktrees. (2017).

[5] Velodyne Lidar. 2016. Velodyne Lidar. https://velodynelidar.com/. (2016).

[6] Maxim Raya and Jean-Pierre Hubaux. 2007. Securing vehicular ad hoc networks. *Journal of Computer Security* 15, 1 (2007), 39–68. http://content.iospress.com/articles/journal-of-computer-security/jcs275

[7] Jason Read. 2017. CDN Performance Summary 2011-2014. http://blog.cloudharmony.com/2014/06/cdn-performance-2011-2014.html. (2017).

[8] M. Szarvas, U. Sakai, and J. Ogata. 2006. Real-time Pedestrian Detection Using LIDAR and Convolutional Neural Networks. In *Intelligent Vehicles Symposium, 2006 IEEE*. 213–218. https://doi.org/10.1109/IVS.2006.1689630

[1] The simulation was written in C and run on a 2015 MacBook Pro containing a 2.5GHz Intel Core i7

[2] We assume 100 byte packets which include 64 byte signatures, 8 byte addresses, and reserve the rest for routing information

[3] The items in parenthesis signify operations that are carried out in parallel