A Decentralized Federated Learning Approach For Connected Autonomous Vehicles

Shiva Raj Pokhrel and Jinho Choi
Deakin University, Geelong, Australia
shiva.pokhrel@deakin.edu.au, jinho.choi@deakin.edu.au

***Abstract*—In this paper, we propose an autonomous blockchain-based federated learning (BFL) design for privacy-aware and efficient vehicular communication networking, where local on-vehicle machine learning (oVML) model updates are exchanged and verified in a distributed fashion. BFL enables on-vehicle machine learning without any centralized training data or coordination by utilizing the consensus mechanism of the blockchain. Relying on a renewal reward approach, we develop a mathematical framework that features the controllable network and BFL parameters, such as the retransmission limit, block size, block arrival rate, and the frame sizes, so as to capture their impact on the system-level performance. More importantly, our rigorous analysis of oVML system dynamics quantifies the end-to-end delay with BFL, which provides important insights into deriving optimal block arrival rate by considering communication and consensus delays.**

***Index Terms*—on-Vehicle Machine Learning, Federated learning, Blockchain, Delay Analysis, consensus delay, low delay**

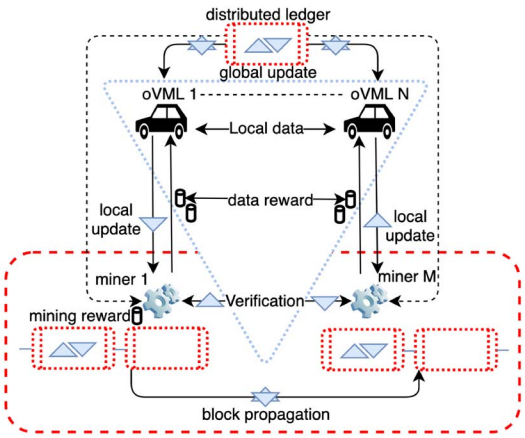

Fig. 1. Interaction between components of our blockchain-based federated learning approach for autonomous vehicles.

I. Background

Next-generation wireless networks are envisaged to guarantee low delay and ultra-high reliability anywhere, anytime and on-the-move [1]–[3]. This will satisfy the real-time communication constraints for the impending autonomous vehicles. To this end, on-Vehicle Machine Learning (oVML) is a persuasive solution wherein each vehicle maintains their best machine learning model and is thereby capable of making intelligent decisions, even when it loses connectivity for some time. Training such an oVML model requires more samples of data than the collected samples by each vehicle. As a result, it demands data trading (and knowledge exchanges) with neighboring vehicles [4], [5]. In this paper, we address the challenge of training each oVML model by exploiting federated learning with their neighboring vehicles [6]–[10].

One main problem is that locally collected data samples are owned by each vehicle. Thus, their trading and knowledge sharing should keep the raw data private from other neighboring vehicles. In this regard, as proposed by Google's federated learning (GFL) [7], each vehicle trades its locally trained model update, (mainly, gradient parameters and learning weightage) rather than the raw data. Due to the potential high number of autonomous cars and the need for them to quickly respond to real world situations, existing cloud-based approach may generate safety risks. Federated learning can represent a solution for limiting volume of data transfer and accelerating the learning processes. In addition, it is worth noting that the regeneration of the raw data from the traded model is not possible, thus guaranteeing privacy. Such trading in GFL is handled by the aid of a centralized server that produces a global model, which is an ensemble average of all the locally trained model updates. Thereafter, each vehicle downloads the globally updated model and computes their next local update until the completion of the global model training process. Observe that, because of the closed-loop exchanges (locally trained model update followed by a globally aggregated model update triggering next iteration of local training), the delay incurred in training completion of GFL may sometimes be around several minutes (10 or more), as reported recently for Google's keyboard application [11].

To this end, we propose (and evaluate) a blockchain-based federated learning (BFL) model, as illustrated in Figure 1, for efficient communication of autonomous vehicles as GFL is not straightforwardly applicable because GFL depends on a single global server, which is vulnerable to server's malfunctioning, highly dependent on network connectivities and suffers heavily due to bottleneck.

To resolve the aforementioned two limitations of GFL, we leverage blockchain [12]–[18] (*remove the centralized global server of GFL and use a blockchain*) and propose a *blockchain-based federated learning* (BFL) approach, where the network system enables exchanging local model updates from vehicles while providing and verifying their corresponding rewards [2],

978-1-7281-5178-6/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:55 UTC from IEEE Xplore. Restrictions apply.

[19]. Figure 1 illustrates our proposed BFL, which outperforms GFL in the autonomous vehicular network system by overcoming the centralized malfunctioning problem and extending the range of its federation to untrustworthy vehicles in a public network thanks to a validation process of the local training modules. More importantly, by providing rewards proportional to the usefulness of data sample sizes, BFL promotes vehicles with a larger size of data samples. An abstract view of the proposed BFL framework consisting of autonomous vehicles and miners is shown as in Figure 1. The miners can physically be either be moving vehicles or separate nodes at network edges (such as WiFi access points or cellular base stations), which are relatively computationally powerful for the *mining process*.[1] The interactions in Figure 1 are explained as follows. Each oVML performs local learning and sends the local model update to its associated miner in the network. All miners exchange and verify all of their local model updates, and then execute their *proof-of-work* [20]. When a miner finishes its *proof-of-work*, it generates a block by recording the validated local model updates. In the end, the block (generated block consisting of the aggregate local model updates) is inserted into the *distributed ledger* of the blockchain. This newly inserted block can then be used by all associated vehicles to compute the required global model update.

With BFL, observe that the global model update is locally computable at each vehicle. Moreover, our BFL design also guarantees that the malfunctioning or failure of a miner, node or a vehicle does have no adverse impact on the global model updates for any other vehicles (overall network system). However, there is a tradeoff: in order to grasp all of the aforementioned benefits, in contrast to the GFL, BFL incurs an additional delay due to the blockchain management in the system. To quantify (and address) the carried-over the delay by blockchain, we conduct the aggregate end-to-end delay analysis of the system with BFL by accounting the *proof-of-work*, communication and computation delays. With relevant insights from the model, we minimize the overall perceived delay of the system by dynamically adjusting the block arrival rate, i.e., the complexity of *proof-of-work* for the blockchain system.

II. Federated Learning: Problem and Solution

Consider $n = 1, 2, \ldots, N$ be the index of autonomous vehicles in a network system and let the n^{th} vehicle collects a set of data samples, s_n and computes its local learning update. The local learning update of the n^{th} vehicle is sent to the associated miner m that is randomly (uniformly) selected from a set $m \in \{1, 2, \ldots, M\}$. Here, N and M represent the numbers of vehicles and miners, respectively. Our federated training is a regression problem that focuses on solving the problem in parallel by considering the entire state space of data samples, $S = \cup_{n=1}^{N} s_n$ collected by all vehicles in our network system. For a global vector v, our objective is as follows:

$$\text{Minimize } \mathcal{F}(v), \tag{1}$$

where

$$\mathcal{F}(v) = \frac{\sum_{n=1}^{N} \sum_{d_j \in s_n} (a_j^{\intercal} v - b_j)^2/2}{|S|} \tag{2}$$

for convenience, $d_j = \{a_j, b_j\} \in S$ denote the j^{th} data sample, where a_j and b_j are the k-dimensional column vector and a scalar respectively.

With GFL, it is well-known that the default idea to solve Equation (1) is to perform local training at each oVML by using the stochastic gradient algorithm, followed by a global training for aggregating local updates via distributed Newton's method. In each stage, the oVML local model is recomputed with the number I iterations. Given the step-size $\delta > 0$, the local vector denoted by v_n is updated after every iteration as follows:

$$v_n^{t+1} = v_n^t - \frac{\delta}{I}\Big(\big(\nabla \mathcal{F}_j(v_n^t) - \nabla \mathcal{F}_j(v)\big) + \nabla \mathcal{F}(v)\Big), \tag{3}$$
$$\text{where } \mathcal{F}_j(v) = \frac{(a_j^{\intercal} v - b_j)^2}{2}.$$

Every oVML in GFL sends $(v_n, \nabla \mathcal{F}_j(v))$, so called local update, to the centralized global server, which then computes the global update, i.e. $(v, \nabla \mathcal{F}(v))$. However, in our proposed GFL framework, the global server is replaced by a blockchain mechanism which is explained below.

A. Decentralized Solution with Blockchain

With BFL, we design a common trustworthy framework for sharing local model updates from all oVML of vehicles via a distributed ledger, where the blocks hold local updates and their validation is performed by using M miners (recall Figure 1). More specifically, each block in the ledger consists of two sections, namely *header* and *body*. The header of a block contains three important parameters: i) *pointer to previous block*, ii) *block arrival rate (denoted by Λ) in blocks per seconds*, and iii) *outcome of the proof-of-work*; and its body carries i) local computational delay and ii) the local update $(v_n, \nabla \mathcal{F}_j(v))$. Each miner maintains a contender block for dumping local updates from all the associated oVMLs and other contending miners. This dumping process carries on and halts either by timeout, τ_{out}, or once the block is full (occupancy reaches maximum block size). Thereafter, the miner generates (randomly) the hash value iteratively by changing its inputs along the lines of that of *proof-of-work*, which terminates only after the final iteration guaranteeing that the generated size of the ultimate hash satisfies the desired target value. More importantly, the contender block is allowed to be a new block in the blockchain system only when the miner successfully generates the hash. This is immediately followed by the propagation of the newly generated block into the system, which acknowledges all other miners and mandates them to i) stop their *proof-of-work* computation

[1]Mining in blockchain is a race between all the contending miners to generate (and validate) a block to append to the blockchain. To have a high likelihood of winning this mining race, it requires substantial computational resources, and the reward for doing so is the mining reward.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:55 UTC from IEEE Xplore. Restrictions apply.

and ii) add the new block to their ledgers. However, there is always a likelihood that in the mean time (before receiving the propagated block) another miner may also generates a block, causing updates in the ledger of its neighboring miners with the later generated block. Such unwanted consequences due to block propagation delay is know as *forking* in the blockchain system [21]. With BFL, forking creates a situation where some oVMLs unknowingly utilize wrong global update to their next iteration of local model update.

Based on the above discussion, it is worth noting that the block arrival rate denoted by Λ depends on the difficulty level of the corresponding *proof-of-work* mechanism, i.e., Λ is controllable by tuning the *proof-of-work* appositely. For example, the lower the desired block arrival rate $\Lambda^{\star}$ is, the smaller the desired target value for hash becomes.[2] More importantly, we have now observed that the occurrence of forking depends mainly on two factors: i) block arrival rate (Λ), ii) block propagation delay, in particular, the probability of occurrence of forking increases with Λ and propagation delay, which is investigated in detail in Sections III and IV.

B. BFL Algorithms Design

Based on the above discussion, we develop two algorithms: i) oVML algorithm at the vehicle; and ii) algorithm at the miner as shown in Algorithm 1 and Algorithm 2 respectively.

Algorithm 1 oVML Algorithm at Vehicle

procedure UPDATE LOCAL MODEL
  for $(i = 0; i \leq I; i++)$ **do**
vehicle n computes Equation (3)
  end for
end procedure
procedure UPLOAD LOCAL MODEL
associate vehicle n with miner m (uniform random)
vehicle n uploads local updates and local computation time
end procedure
procedure DOWNLOAD GLOBAL MODEL
vehicle n downloads new block from its miner m
end procedure
procedure UPDATE GLOBAL MODEL
  while $((v(t+1) - v(t))^2 > tol)$ **do**
compute global vector $v(t+1) \leftarrow v(t) + \sum_n \frac{I}{|S|}(v_n^t - v(t))$
  end while
end procedure

Recall that GFL is highly vulnerable to the malfunctioning of centralized server which may cause distortion to the global models of all vehicles. In contrast, our proposed BFL computes global model update locally at each vehicle, which not only increases robustness against malfunctioning of the centralized server but also i) reduces the computational delay and ii) eliminates the global update propagation delay completely (from the centralized server to vehicles).

[2]We use the *proof-of-work* for tractability, however, use of other consensus algorithms such as *Byzantine-fault tolerance* and *proof-of-stake* into BFL is quite straightforward, which may require more sophisticated computations and preambles to arrive at a consensus among miners.

Algorithm 2 Algorithm at Miner

procedure CROSS VERIFICATION
miner m broadcasts the obtained local model update
  while (block size $<$ max. size) or $(timer < \tau_{\text{out}})$ **do**
    if (verify local update) **then**
Insert local update into contender block
block size$\leftarrow$ block size + size (local update)
    end if
  end while
end procedure
procedure GENERATE BLOCK
  while (hash satisfies target) or (receives generated block) **do**
execute proof-of-work
    if (hash satisfies target) **then**
create new block(contender block)
      if (receives generated block) **then**
        if (forking) **then**
send acknowledgments.
        end if
      end if
    end if
  end while
end procedure
procedure PROPAGATE BLOCK
broadcast new block
  procedure AVOID FORKING
wait for acknowledgments (time out after τ_{out})
  end procedure
end procedure

III. BFL SYSTEM ANALYSIS

We consider two assumptions to make our analysis tractable.

ASSUMPTION 1. The computation and verification duration inside oVML is sufficiently small as compared to communication delay in our proposed BFL system.

As computationally powerful servers are installed inside the vehicles, it is reasonable to assume that the wireless communication channel is the bottleneck for the proposed BFL system, and the local and global learning times inside vehicles are typically negligible.

ASSUMPTION 2. The *proof-of-work* follows a Poisson process.

Note that Assumption 2 is a standard assumption that have been used in [18], [22]–[25].

ASSUMPTION 3. All miners are synchronized with each other and start their *proof-of-work* at once by maintaining $timeouts$.

A. Modeling Wireless Path

Let Θ denote the maximum rate at which the channel can transmit the link layer frames to the vehicle, (i.e., Θ =*bit rate÷Frame size in bits*). Let v be the speed of the vehicle, and f_c the carrier frequency, the Doppler frequency is $f_d = f_c v/c$ ($c = 3 \times 10^8$ m/s). Considering F as the fading margin and given the physical modulation and coding schemes, the channel is in the poor state, if received Signal to Noise Ratio (SNR) is below a threshold $\mathbb{E}[\text{SNR}]/F$,; else, it is in the ideal state. The average probability that a frame transmission fails

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:55 UTC from IEEE Xplore. Restrictions apply.

due to errors in the channel is[3] $\bar{p}_e = 1 - e^{-1/F}$. Representing $\eta = \sqrt{2/(F(1-\rho^2))}$, $\rho = J_0(2\pi f_d/\Theta)$, with, ρ is the Gaussian correlation coefficient of two samples of the amplitude of a fading channel with frequency f_d; $1/\Theta$ is the frame transmission time over the channel; and $J_0(.)$ is the zero order Bessel function. The stationary state transition probabilities with the Marcum-function $\mathbb{Q}(.,.)$ are as follows:

$$p_p = \boldsymbol{Pr}(\text{Remains in poor}) = 1 - \frac{\mathbb{Q}(\eta, \rho\eta) - \mathbb{Q}(\rho\eta, \eta)}{e^{\frac{1}{F}} - 1},$$
$$p_i = \boldsymbol{Pr}(\text{Remains in ideal}) = \frac{1 - p_{e,\text{Cell}}(2 - p_p)}{1 - \bar{p}_e}.$$

Let L be the required number of link layer frames for transmitting a block or a local update in the cellular channel, then

$$L_{\text{block/update}} = \frac{\text{Block/Local update size in bits}}{\text{Frame size in bits}}.$$

With the Markovian process embedded at the starts of frame transmissions, we denote ℓ_i as the probability that at least one frame fails out of i link layer transmissions, when the channel state is $ideal$; and $\nu_i^{(r)}$ the probability that at least one fails out of i link layer transmissions, given that the first link layer frame has already had r (with $r < R$, here, R is the maximum number of (re)transmissions) transmission attempts failures while the channel state is at $poor$. The renewal equations to estimate these probabilities are

$$\begin{aligned} \ell_i &= p_i \ell_{i-1} + (1 - p_i)\nu_{i-1}^{(0)} \\ \nu_i^{(r)} &= (1 - p_p)\ell_i + p_p \nu_i^{(r+1)}. \end{aligned}$$

Here, ℓ_i, $1 \le i \le L$, and $\nu_i^{(r)}$, $0 \le r \le R$, are obtained by imposing the conditions

$$\ell_1 = 0, \ \nu_1^{(r)} = {p_p}^{(R-r)}, \ \text{and} \ \nu_i^{(R)} = 1.$$

A block (or an update) is discarded in the channel with the following probability:

$$p_d = \nu_L^{(0)} \frac{(1 - p_i)}{(2 - p_p - p_i)} + \frac{\ell_L}{1 + \frac{(1-p_i)}{(1-p_p)}}. \tag{4}$$

The effective block/update rate over the channel, accounting the link layer (re)transmissions, can be computed with the following set of renewal equations:

$$\begin{aligned} u_i &= 1 + p_i u_{i-1} + (1 - p_i)\varrho_{i-1}^{(0)} \\ \varrho_i^{(r)} &= 1 + p_p \varrho_i^{(r+1)} + (1 - p_p)u_i \\ \varrho_i^{(R)} &= 1 + p_p \varrho_{i-1}^{(0)} + (1 - p_p)u_{i-1} \end{aligned}$$

with boundary conditions $u_1 = 1$ and $\varrho_1^{(R)} = 1$. Here, i) given that the initial channel state is $ideal$, u_i is the average link layer frames needed to be transmitted for a block/update consisting of i link frames; ii) $\varrho_i^{(r)}$ is the average number of link layer frames needed to be transmitted for a block/update given that the first link layer frame has already had r failed transmission attempts and the initial state is $poor$.

Finally, the expected number of frames needed to be transmitted for a block/update consisting of L frames is

$$u_{\text{block/update}} = \frac{u_L(1 - p_p)}{(2 - p_p - p_i)} + \frac{\varrho_L^{(0)}}{1 + \frac{(1-p_p)}{(1-p_i)}}. \tag{5}$$

Based on the above analyses, with slight abuse of notation and using our proposed framework (see Figure 1), the communication delay experienced by the vehicles while uploading the local updates and downloading global model (i.e., a block from the distributed blockchain ledger) can be estimated as

$$\tau_{\text{lup}} = \frac{u_{\text{update}}}{\Theta} \ \text{and} \ \tau_{\text{gdn}} = \frac{u_{\text{block}}}{\Theta} \tag{6}$$

respectively.

B. Modeling Blockchain Delay

As illustrated in Algorithm 2, the blockchain delay consists of block arrival, propagation and verification delays. We assume that the verification time is sufficiently small (almost negligible) when compared to the communication delays. Therefore, considering m contending miners the total cross verification delay (steps 1-9, Algorithm 2), denoted by τ_v, of the system can be found as

$$\tau_v = \max\Big\{(\tau_{\text{out}} - \tau_{\text{lup}}), \sum_m \frac{u_{m_{\text{update}}}}{\Theta}\Big\} \tag{7}$$

where $u_{m-\text{update}}$ is the expected number of frames needed to be transmitted from miner m. Denote by $\mathbb{E}[SNR_m]$ the perceived expected SNR by miner m while receiving frames from other miners, which is required for computing u_m-*update* (along the same lines of (5)).

Similarly, the total block propagation delay, denoted by τ_p, from the winning miner $\hat{m}$ (step 22, Algorithm 2) is given by

$$\tau_p = \max\Big\{\tau_{\text{out}}, \sum_m \frac{u_{\hat{m}-\text{block}}}{\Theta}\Big\}, \tag{8}$$

where $u_{\hat{m}-\text{block}}$ is the expected number of frames needed to be transmitted from the miner $\hat{m}$ (and can be computed along the lines of (5)).

Finally, the block arrival delay (steps 10-21, Algorithm 2), denoted by $\mathcal{T}_g$, can be estimated as follows. Using Assumption 2, the block arrival delay can be modeled by a random variable with an exponential distribution. By fitting the distribution of block arrival to the exponential distribution, the mean or expected value of such an exponentially distributed random variable (i.e., the block arrival delay) with block arrival rate parameter Λ is given by (using (4) and (9))

$$\mathcal{T}_g = \big((1 - p_d)\Lambda\big)^{-1}. \tag{9}$$

Observe that, using (9), the delay of interest in this case is the block arrival delay of the winning miner, given that the block is not discarded with probability p_d in the channel.

Finally, we compute the expected overall system delay, denoted by $\mathbb{E}[T]$ by, using a renewal reward approach. With

[3]We assume that the underlying channel dynamics changes between an ideal state (in which all transmissions are successful) and a poor state (in which all transmissions are failed).

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:55 UTC from IEEE Xplore. Restrictions apply.

n vehicles and m miners, the expectation $\mathbb{E}_{(n,m)}[T]$ can be computed as

$$\mathbb{E}_{(n,m)}[T] = \big(\tau_{\text{local}} + \tau_{\text{lup}} + \tau_v + \tau_p + \mathcal{T}_g + \tau_{\text{gdn}} + \tau_{\text{global}}\big) + p_{\text{fork}}\big(\mathbb{E}_{(n,m)}[T] - \tau_{\text{gdn}} - \tau_{\text{global}}\big), \quad (10)$$

where τ_{local} and τ_{global} are the times involved in the local (steps 1-4, Algorithm 1) and global (steps 13-17, Algorithm 1) updates of the models, respectively. The forking probability p_{fork} required in (10) can be computed as

$$p_{\text{fork}} = 1 - \prod_{m \neq \hat{m}} \boldsymbol{Pr}\big((\tau_m - \tau_{\hat{m}}) > \tau_p\big), \quad (11)$$

where $\tau_m = (\tau_{\text{local}} + \tau_{\text{lup}} + \tau_v + \mathcal{T}_g)$ is the time delay before miner m generates a block and $\tau_{\hat{m}}$ is that of the winning miner.

C. Minimizing Delay for Autonomous Vehicles

With relevant insights from the derived expected delay of the system (see (8)), we propose an approach to evaluate the optimal $\Lambda^\star$ that minimizes the delay of the involved *proof-of-work* process. Observe that the *proof-of-work* process affects i) block arrival/generation delay ($\mathcal{T}_g$), ii) block propagation delay (τ_p), and iii) forking probability (p_{fork}), all of which are interlinked with each other due to the winning miner. Next, we have the following approximation.

Based on Assumption 3, all miners are synchronised with each other and start their *proof-of-work* process at once by maintaining τ_{out} such that $\tau_v = \tau_{\text{out}} - (\tau_{local} + \tau_{lup})$.

Assumption 3 also mandates that all miners wait till τ_{out}, even after completing the verification (steps 1-9, Algorithm 2), which provides the following approximation:

$$\mathbb{E}_{(n,m)}[T] \approx \hat{T}_{n,m} = \frac{\big(\tau_{\text{out}} + \mathbb{E}[\mathcal{T}_g]\big)}{1 - p_{\text{fork}}} + \tau_{\text{global}} + \tau_{\text{gdn}}. \quad (12)$$

IV. Performance Evaluation

In this section, we evaluate the overall performance of the proposed system and gain important insights on the overall learning completion time for the system under different channel conditions, by conducting numerical and simulation experiments. The network setting and parameters used for our evaluation are based on 3GPP LTE Cat. M1 specification.[4]

Figure 4 illustrates the impact on the overall system delay with respect to block arrival rate, i.e. demonstrates a snapshot of $\mathbb{E}_{n,m}[T]$ vs. Λ (recall the closed form approximation derived in Equation (22)). As expected, the delay decreases with increase in SNR, which is captured well by both numerical and simulation results. Furthermore, Figure 4 demonstrates that, for given network setting (in our experiment the attainable value of Λ under the given network condition of Figure 4 is $0.02 < \Lambda < 0.2$), the function, $f(\Lambda)$, is convex and its epigraph (the set of points on or above the graph of $f(\Lambda)$) is a convex set. In fact, it is known that if a twice-differentiable function of a single variable is convex, then its second derivative must be non-negative on its entire domain. This provides us with an important observation as follows:

O$_1$) The minimization of $\mathbb{E}_{n,m}[T]$ can be achieved by adjusting Λ appropriately as the achieved local minimum of such a convex function is also a global minimum.[5]

Next, we evaluate the impact of wireless transmission failures on performance due to fading. Figure 5 depicts the probability that a block or a local update is discarded (due to repeated failures) in the channel. It also depicts the probability of forking with increase in transmission failures due to fading. It can be observed that both p_d and p_{fork} increase approximately log linearly, i.e., increase exponentially with the increase in transmission failures in the channel p_e; p_d is slightly lower than p_{fork}. Our analytic modelling accurately captures these facts.

Since the increase in wireless transmission failures (increases discards, recall Equation (9)), on average, requires a higher number of retransmissions, before successful transmission, which increases the service time of a frame in the channel. The increase in delay provides more time for contending miners to complete their block generation process (recall Equations (16) and (19)). Hence, our second main observation is as follows:

O$_2$) With increasing transmission failures, *the increase on forking probability is higher than on* p_d, which adversely effects performance and causes substantial increase in the overall delay of our BFL system.

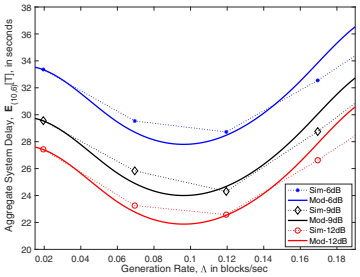

Fig. 2. Mean system delay with increase in block arrival delay under different channel conditions.

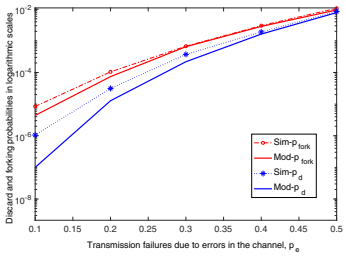

Fig. 3. Forking and discard probabilities under different channel conditions.

[4]see technical reports at https://www.3gpp.org/DynaReport.

[5]It is worth noting that a strictly convex function will have at most one global minimum however precise determination such strict convexity without Approximation 1 requires further investigations and is left for future work.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:55 UTC from IEEE Xplore. Restrictions apply.

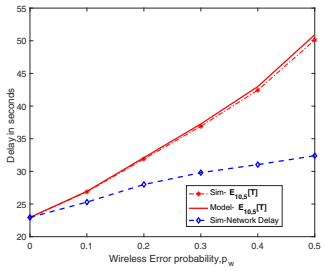

Fig. 4. Impact of channel errors on the overall system and network delay.

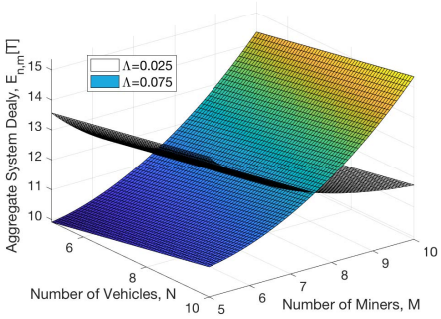

Fig. 5. Mean system delay with variation in number of miners and vehicles for two different block arrival rates (0.025 and 0.075 blocks/sec) under static channel conditions.

With a low probability of transmission failures (say $p_e \leq 0.1$), the small loss is masked by the R (re)transmissions in that the discard probabilities are almost zero. With significant wireless channel errors, however, the $K = 7$ retransmissions are not enough to mask the repeated failures, and blocks are lost due to frame discards, which again leads to increase in the forking probability (Figure 5).

V. Conclusive Remarks

In this paper, we have enhanced federated learning with blockhain for the performance and privacy of autonomous vehicles. Our BFL framework facilitates efficient communication of autonomous vehicles, where local on-vehicle learning modules exchange and verify their updates in a fully decentralized fashion. BFL has successfully enabled on-vehicle machine learning without any centralized coordination by exploiting the consensus mechanism of blockchain. Furthermore, we developed a comprehensive analysis of BFL network system dynamics for the end-to-end system delay, which provides important insights for deriving optimal block arrival rate.

References

[1] D. Feng, C. She, K. Ying, L. Lai, Z. Hou, T. Q. Quek, Y. Li, and B. Vucetic, "Toward ultrareliable low-latency communications: Typical scenarios, possible solutions, and open issues," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 94–102, 2019.

[2] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, 2019.

[3] Y. Sun, L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran, "Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment," *IEEE Internet of Things Journal*, 2019.

[4] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[5] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *arXiv preprint arXiv:1909.11875*, 2019.

[6] W. Chen, Z. Zhang, Z. Hong, C. Chen, J. Wu, S. Maharjan, Z. Zheng, and Y. Zhang, "Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things," *IEEE Internet of Things Journal*, 2019.

[7] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[8] S. Samarakoon, M. Bennis, W. Saady, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *arXiv preprint arXiv:1807.08127*, 2018.

[9] J. Park, S. Wang, A. Elgabli, S. Oh, E. Jeong, H. Cha, H. Kim, S.-L. Kim, and M. Bennis, "Distilling On-Device Intelligence at the Network Edge," *arXiv preprint arXiv:1908.05895*, 2019.

[10] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A Joint Learning and Communications Framework for Federated Learning over Wireless Networks," *arXiv preprint arXiv:1909.07972*, 2019.

[11] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated Learning for Wireless Communications: Motivation, Opportunities and Challenges," *arXiv preprint arXiv:1908.06847*, 2019.

[12] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[13] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *IEEE P2P 2013 Proceedings*. IEEE, 2013, pp. 1–10.

[14] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, 2019.

[15] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[16] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[17] H. Seo, J. Park, M. Bennis, and W. Choi, "Communication and consensus co-design for low-latency and reliable industrial *IoT* systems," *arXiv preprint arXiv:1907.08116*, 2019.

[18] R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Block arrivals in the Bitcoin blockchain," *arXiv preprint arXiv:1801.07447*, 2018.

[19] M. Shen, X. Tang, L. Zhu, X. Du, and M. Guizani, "Privacy-preserving support vector machine training over blockchain-based encrypted *IoT* data in smart cities," *IEEE Internet of Things Journal*, 2019.

[20] G. Kumar, R. Saha, M. K. Rai, R. Thomas, and T.-H. Kim, "Proof-of-Work consensus approach in Blockchain Technology for Cloud and Fog Computing using Maximization-Factorization Statistics," *IEEE Internet of Things Journal*, 2019.

[21] S. Wang, C. Wang, and Q. Hu, "Corking by Forking: Vulnerability Analysis of Blockchain," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 829–837.

[22] J. Göbel, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor, "Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay," *Performance Evaluation*, vol. 104, pp. 23–41, 2016.

[23] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, 2014.

[24] ——, "Analysis of Bitcoin pooled mining reward systems," *arXiv preprint arXiv:1112.4980*, 2016.

[25] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

Authorized licensed use limited to: Politecnico di Milano. Downloaded on September 17,2020 at 13:14:55 UTC from IEEE Xplore. Restrictions apply.