A Prediction-Enhanced Physical-to-Virtual Twin Connectivity Framework for Human Digital Twin

Samuel Okegbile, Jun Cai, Junjie Wu, Jiayuan Chen, and Changyan Yi

Affiliation not available

December 22, 2023
A Prediction-Enhanced Physical-to-Virtual Twin Connectivity Framework for Human Digital Twin

Samuel D. Okegbile, Member, IEEE, Jun Cai, Senior Member, IEEE, Junjie Wu, Jiayuan Chen, and Changyan Yi Member, IEEE

Abstract—This paper proposes a new secure and privacy-preserving prediction-enhanced solution for reliable physical-to-virtual communications in human digital twin (HDT) systems. With such a prediction-enhanced connectivity (PeHDT) framework, the evolution of any virtual twin (VT) could be triggered in real-time or in advance using the expected state of its physical counterpart. This ensures the continuous maintenance of a true replica of each physical twin (PT), thus relieving the need for timely PT-VT synchronization while the VT-experienced delay is reduced to zero or close to zero. We adopted a secured federated multi-task learning technique to meet the security and privacy constraints of HDT and employed a single server discrete-time batch-service queue framework when characterizing the batching process to reduce the communication burden. Furthermore, we introduced a prediction verification framework to improve the performance of the proposed PeHDT framework. The resulting problem was formulated as a constrained Markov decision process and was solved by introducing a primary-dual deep deterministic policy gradient (DDPG) algorithm. Through a joint investigation of communication, batching and prediction verification schemes, the simulation results show that the proposed PeHDT framework can greatly reduce both the VT-experienced delay and the PT-VT communication time without compromising the specific requirements of HDT.

Index Terms—Batching process; constrained Markov decision process; federated multi-task learning; human digital twin; physical-to-virtual twin connectivity; prediction; primary-dual DDPG.

1 INTRODUCTION

HUMAN digital twin (HDT) is an innovative technology that seeks to represent humans in the virtual world. It aims to create a digital replica of each human, located in the physical world, to ensure the development and reuse of various human-centric solutions. Such a digital replica, also called a virtual twin (VT), is expected to reflect the static and dynamic characteristics of its counterpart human, otherwise referred to as a physical twin (PT), with the ability to optimize the decision-making process and the overall performance in the physical environment (PE). One key requirement of this HDT includes providing VT, in real-time, an ability to accurately predict the current and future states of its corresponding PT, based on its learning experience and available historical states or data [1], thus facilitating efficient suggestions, recommendations or information required to optimize the performance or the state of the PT in the PE.

To improve the learning experience and increase the available historical states in the virtual environment (VE), a timely PT-VT synchronization scheme has been discussed as a useful solution [2]–[4]. However, such a solution may suffer from high communication costs owing to transmission delay and packet loss probability, causing a high synchronization gap. Hence, maintaining an updated VT in real-time, via the standard PT-VT connectivity scheme is very challenging if the HDT-specific latency and reliability (e.g., packet loss probability) requirements of 1 ms and $10^{-6}$, respectively as well as its security and privacy constraints must be satisfied. To realize a reliable PT-VT synchronization, we consider a prediction-enhanced PT-VT connectivity scheme where the evolution (i.e., updating) of any typical VT is triggered in real-time (or in advance) based on the expected state of its counterpart PT. With such an approach, the VT aims to produce a true replica at every time slot by predicting the state of the PT based on its previous historical states and experience. As a result, the VT is maintained in real-time or even in advance to relieve the need for timely PT-VT synchronization while the VT-experienced delay could be reduced to zero or close to zero.

However, such a prediction-based connectivity scheme can suffer from a high prediction error since predictions are generally not error-free [5] and rely on available historical states and VT learning experience. As a result, it becomes important to investigate how to limit the impacts of prediction errors on the overall performance of the HDT system thereby improving its reliability. To achieve this, we incorporate a prediction verification system where the actual states received from the PT are used to verify the predicted states at the VT. This ensures that predictions with errors are later replaced by the actual states when the error margin is more than a predefined threshold while contributing to the VT learning experience. Since predictions are made by the VT, the need for timely and continuous PT-VT communications (which is indeed very challenging due to constraints on energy availability) can be relaxed,
which introduces a possibility to further reduce the communication costs between any PT-VT pair. Leveraging this, we introduce an update batching approach where new data generated by each PT are transmitted in batches. Suppose that a predictor, located in the VE, can estimate the expected PT states in a longer prediction horizon; in that case, the timely evolution of VT is less sensitive to packet losses in communications, and the batch formation rate can be reduced to lower the communication burden.

While the introduction of an update batching approach can reduce the communication burden, such an approach may complicate the prediction verification time – defined as the total time between the time any typical prediction was made to the time its corresponding actual data were received from the PT. More specifically, every prediction should be verified within the acceptable verification time threshold to meet users’ quality of experience requirements [6]. This prediction verification time depends on the PT-VT synchronization delay — defined as the total time from the generation of a status update by any PT to the time such a status update is received by the corresponding VT — and the prediction horizon [7], generally defined as how far ahead the VT predicts the future. Thus, the prediction horizon and the PT-VT synchronization delay must be carefully studied to meet the verification time requirement. Interestingly, a higher prediction horizon provides more benefits to users since the VT can predict the future thereby facilitating improved decision making in the PE.

It immediately becomes necessary to answer the following questions: 1) how do we determine the optimal batch size or the batch formation rate that can minimize the communication costs without compromising the overall reliability of such a prediction-enhanced system? 2) given the need to determine the future state of any PT, what is the best prediction horizon that enhances the overall performance of the system? 3) how to improve transmission reliability through reduced transmission loss probability and PT-VT synchronization delay? 4) and subject to the energy availability constraints, what is the best PT-VT synchronization delay that optimizes the prediction verification time in the VE?

### 1.1 Motivation and Contribution

Indeed, providing answers to the questions raised above is not straightforward since an increase in the batch size reduces the batch formation rate while complicating the predictions’ verification time (i.e., increases the predictions’ verification time for the early arriving packets within a batch and possibly decreases the predictions’ verification time for the late arriving packets within a batch). Alternatively, under some special conditions where the batch processing rate may be much higher (i.e., very low PT-VT synchronization delay) than the batch formation rate, an increase in the batch formation rate may reduce the prediction verification time at the expense of a higher communication burden. Obtaining stable batch processing under such stochastic processes is, therefore, challenging. In addition, an increase in the prediction horizon improves the timeliness of VT at the expense of a higher prediction error while a PT-VT synchronization delay increases the predictions’ verification time. A trade-off, therefore, exists between i) overall reliability (including prediction reliability and connectivity reliability) and PT-VT synchronization delay (and thus the communication costs), ii) prediction horizon and prediction error, as well as iii) predictions’ verification time and prediction horizon. This makes the proposed prediction-enhanced connectivity scheme very complicated to analyze.

To address the challenges impacting the proposed prediction-based connectivity scheme, we introduce an update batching-based synchronization process where newly arriving updates are batched for group processing subject to energy availability while satisfying the overall system reliability requirement. The batching process ensures that packets are only processed for subsequent offloading to the VE when there is sufficient energy to guarantee successful operations at the LA. To the best of our knowledge, such a work that considers the PT-VT connectivity problems in HDT through prediction-enhanced and update batching-enabled techniques has never been studied. The main contributions of this paper are summarized as follows:

- We first propose a prediction-enhanced, secure and privacy-preserving PT-VT connectivity scheme for HDT framework (PeHDT) to satisfy the stringent specific requirements of HDT while facilitating reliable future state predictions of any PT via its paired VT. In addition, a secured federated multi-task learning (FML) technique was integrated via the integration of blockchain and FML techniques to meet the security, model accuracy and privacy requirements of HDT.
- To lower the communication burden when verifying predictions in the proposed PeHDT, an update batching queuing method was incorporated by characterizing the PT-VT connectivity process as a single server discrete-time batch-service queue with batch size-dependent service. A stochastic geometry-based grant-free power ramping communication technique was then used to improve the PT-VT transmission reliability.
- Next, we derive closed-form expressions for PT-VT synchronization delay, predictions’ verification time and error probability, while demonstrating the associating tradeoff among such metrics in the PeHDT.
- We then formulate the resulting problem as a constrained Markov decision process (cMDP) – a variation of the standard MDP that is more suitable for modeling scenarios with specific requirements – and adopt the primary-dual deep deterministic policy gradient (DDPG) algorithm to provide a solution to the optimization problem.
- Finally, we obtained results through numerical simulations and compared the obtained results with the other related approaches.

This is the first paper that addresses the connectivity problems in HDT through a joint investigation of communication, batching and prediction verification schemes while satisfying the security and privacy requirements of HDT.

### 1.2 Organization

The rest of this paper is structured as follows. Section 2 presents the review of related work. Section 3 introduces
the system model while Section 4 presents the analysis of the PT-VT synchronization delay including the details of the key constraints of the proposed prediction-enabled scheme. In Section 5, we present optimization problems and the details of the adopted technique. Section 6 presents the numerical simulations, while Section 7 concludes the paper and offers suggestions for future works.

2 RELATED WORK
In this section, we present a review of the related work. For clarity, we categorized the reviews into two groups: physical-virtual communications and predictions for communications, before summarizing the uniqueness of this current work.

2.1 Physical-virtual communications
A physical-virtual communication has been identified as a key problem in HDT [1]–[3], where FL and blockchain technologies were introduced as useful tools to realize secure and privacy-preserving PT-VT connectivity. Most especially, the work in [2] presented an edge-assisted human-to-virtual twin connectivity scheme for HDT to prevent data leakage and eavesdropping attacks. The long-term average connectivity cost was minimized to optimize offloading and validation energy consumption. In [3], a secure, privacy-preserving and efficient human-to-virtual twin connectivity scheme for HDT was presented by integrating differential privacy, FML and blockchain. The work demonstrated that the proposed framework can accelerate the learning process without sacrificing much accuracy, privacy and communication costs. Similarly, the authors in [8] designed a digital twin wireless networks model, where digital twin technology was adopted to eliminate unreliable and long-distance communications among end users and edge servers. A permissioned blockchain-empowered FL framework was proposed to facilitate edge computing in the proposed system.

In addition, a digital twin-edge network was proposed in [9], where the running states and behaviour of internet of things devices were modelled through a blockchain-empowered FL scheme for digital twin-edge networks to improve the learning security and privacy of users. A directed acyclic graph and consortium blockchain were exploited in [10] to obtain a digital twin edge networks framework, where the authors adopted FL when constructing digital twins of smart devices. A cooperative FL was proposed to improve the performance of the FL algorithm and local model update verification process. While FL and blockchain have been demonstrated as useful tools to ensure security and privacy in digital twin-based systems, it remains unclear how to meet the extreme end-to-end delay and reliability requirements in such systems since the adoption of such tools can significantly complicate the users experienced latency [1], [11].

2.2 Predictions for communications
Recently, a prediction-enabled solution has been adopted to achieve ultra-reliable and low-latency communications in tactile internet [5], haptic communication [12] and many other application areas. For instance, a long short-term memory (LSTM)-based prediction model with an optimized prediction window size was proposed in [12] to predict future haptic information. The predicted haptic information was generally transmitted in advance according to the window size to satisfy the delay requirement. Similarly, an LSTM and multi-layer perception neural network was adopted in [13] to reduce user-experienced delay through prediction, pre-rendering and caching of virtual reality videos at the intelligence edge server. The authors in [14] also studied the synchronization between devices in the PE and their digital models in the metaverse by presenting a sampling, communication, and prediction co-design scheme to minimize the communication load subject to the tracking error constraint. A knowledge-assisted constrained twin-delayed deep deterministic algorithm was exploited to optimize the sampling rate and the prediction horizon.

Furthermore, the ability of a prediction-based scheme to minimize the processing time in an augmented reality robotic telesurgical application was demonstrated in [15], while a reduced wireless resource utilization was achieved in [16] via a joint optimization of the communication and packetized predictive control system. The lack of wireless resources was overcome in [17] through the proposed Gaussian process regression-based prediction and machine learning-enabled solution. In [18], the authors proposed a task-oriented prediction and communication co-design scheme with reliability formulated as a function of both prediction errors and packet losses, while a prediction-enhanced model was formulated in [19] to improve the real-time vehicular communication link by sending the predicted data to the receiver in advance. A similar solution was presented for the time-critical industrial internet of things applications in [20] to enhance the performance of the age of information for short-packet transmission.

While prediction-based approaches have been demonstrated in several works to be capable of reducing the end-to-end delay of mission-critical applications, most existing solutions deployed the predictors at the transmitters. By deploying the predictors at the VTs (i.e., receivers), we can limit the impact of transmission error and low transmission reliability on the evolution of VTs. In addition, we incorporate security and privacy – two key requirements of HDT – into the proposed framework and formulate a batching process to reduce communication costs. The proposed framework adopted the power ramping random access technique to improve transmission reliability between any typical PT and its paired VT. The common notations used throughout this paper are summarized in Table 1.

3 SYSTEM MODEL
In this section, we present the overall framework as well as the PT-VT synchronization delay and reliability requirements of the proposed PeHDT.

3.1 PeHDT framework
We consider a PeHDT system, as shown in Fig. 1, where each PT, containing \( N \) sensing devices, could generate observations such as blood pressure, genetic components, temperature, location, humidity, etc. Through its local aggregator (LA) and the connecting global aggregator (GA),
the PT sends its data (i.e., its current states) to its counterpart VT to enhance the maintenance of such a VT. In this paper, the considered system consists of multiple PTs (each with its corresponding LA), which are spatially distributed following the Poisson point process (PPP), denoted as $\Phi_{PT}$ with intensity $\mathcal{U}_{PT}$. In practice, the corresponding VTs are commonly stored in edge servers, which are also spatially distributed following PPP, denoted as $\Phi_{VT}$ with intensity $\mathcal{U}_{VT}$ [2], [21]. To minimize the transmission delay, each PT is associated with the nearest edge server for storing its corresponding VT [22], [23]. We discretize time into slots and use the terms packet and state interchangeably to refer to any newly generated PT state.

To prevent potential privacy leakage, we employ a secured FML technique, which integrates FML [3] and blockchain technology so that only model parameters are offloaded to the VE and the quality of each model needs validation during the final communication round of FML to ensure its accuracy. As shown in Fig. 1, the actual states from the PT are first stored in a queue with a finite buffer of size $Q$ and then trained in batches of varying size $B_m$, according to the minimum and maximum threshold limits $B_{\text{min}}$ and $B_{\text{max}} \leq Q$, respectively. After a batch is formed at any typical LA $m \in [M]$, the connecting GA selects $M$ neighbouring LAs (including LA $m$) to participate in the FML. Following the FML, the local model of each typical LA $m$, denoted as $\omega_m$, is divided into a shared model $\omega_m^s$ and task-specific model $\omega_m^t$. The FML aims to improve the task-specific model $\omega_m^t$ of LA $m$ through the aggregation of sharable models $\omega_m^s(i \in \{1, 2, ..., M\})$ from any selected $M$ neighbouring LAs similar to the hard parameter sharing technique [24]. Each selected LA then receives the global model $\omega_{GA}$ from the GA and updates its locally trained model by computing the gradient for both shared and task-specific models. The training of $\omega_m^s$ continues similar to the conventional FL technique. During the final communication round of the FML training, model $\omega_m^s$ of LA $m$ is validated using blockchain. Following the completion of a secured FML process, the LA $m$ will offload the trained model to its paired VT for updating via the power ramping scheme [22].

Clearly, the introduction of a secured FML technique as well as the PT-VT model transmission process introduces PT-VT synchronization delay, a metric that is equivalent to the total time from when a typical packet is generated by any PT to the moment it is successfully received by its corresponding VT. Thus, it is possible that the latency requirement of HDT may not be met, which justifies the need for the prediction on VT. Let $X_m(t) = [x_m^i(t)[i = 1, ..., f]]^T$ capture the packet of any typical $m$th PT in any time slot $t$, with $f$ representing the number of features. We denote the state $X_m(t)$ received by the corresponding VT in the VE as $Y_m(t + D_m^c)$, where $D_m^c$ represents the end-to-end delay, including the number of slots required to process and transmit $X_m(t)$ between such a PT-VT pair. We know that $Y_m(t + D_m^c) = X_m(t)$, provided that the packet that conveys $X_m(t)$ is successfully received in time slot $t + D_m^c$ such that the PT-VT synchronization delay is $D_m^c$. This $D_m^c$ is a random variable in practical systems.

The proposed PeHDT framework can eliminate the VT-experienced delay while reducing PT-VT communication delay. Generally, each VT predicts its corresponding PT’s expected present or future states based on its previous learning experience and historical states. Let the prediction horizon at any time slot $t$ be denoted as $T_m^p$, $0 \leq T_m^p \leq T_m^{\text{max}}$. We consider three different cases.

- Case 1 – Predicting the present state: In this case, there are two possible scenarios. Under the first scenario, the VT always predicts the current PT state $X_m(t)$ regardless of whether the previous actual state $X_m(t - 1)$ has been received or not. Under such a scenario, the prediction for the present time $t$ is made based on either $Y_m(t - 1 + D_m^c) = X_m(t - 1)$, if it is available, or the previous predicted state $\tilde{Y}_m(t - 1) \approx X_m(t - 1)$. Thus, the prediction for time $t$ equals the actual data generated by the PT at time $t$ which is equivalent to the actual data received at time $t + D_m^c$. That is,

$$\tilde{Y}_m(t) \approx X_m(t) = Y_m(t + D_m^c),$$

$$\forall t, D_m^c \geq T_m^p, T_m^p = 1.$$ (1)

However, the prediction horizon may generally depend on the last received actual packets. Thus, the second scenario relies on only the last received actual packets and can be expressed as

$$\tilde{Y}_m(\tau + (t - \tau)) = X_m(t) = Y_m(t + D_m^c),$$

$$\forall \tau, D_m^c \geq T_m^p, T_m^p = t - \tau.$$ (2)

where $\tau$ represents the time the last actual packet was received by the VT such that the prediction for time $t + (t - \tau)$ equals the actual data generated by
the PT at the time $t$ which is equivalent to the actual data received at time $t + D^c_m$. In this case, the VT-experienced delay is given as $D^v_m = 0$. For the sake of consistency, we henceforth focus on predictions based on the last received actual packets as in (2).

- Case 2 – Wait before predicting the latest state: In this case, the VT waits for a short time $D^th_m \geq 0$, before predicting the latest state of its paired PT. Hence, the prediction for time $t - D^th_m$ equals the actual data generated by the PT at time $t - D^th_m$, which is equivalent to the actual data received at time $t + D^c_m - D^th_m$, such that

$$
\tilde{Y}_m(\tau + (t - \tau) - D^th_m) = X_m(t - D^th_m)
$$

$$
= Y_m(t + D^c_m - D^th_m), \forall t, D^c_m > D^th_m, D^c_m \geq T^p_m,
$$

$$
T^p_m = (t - \tau) - D^th_m. \quad (3)
$$

In this case, the VT-experienced delay is given as $D^v_m = D^th_m$. Generally, case 2 is reduced to case 1 when $D^th_m = 0$, while its worst-case is obtained when $D^th_m = D_{req}$, which defines the special case where the VT waits for the time $D_{req}$ before predicting the latest state of its paired PT.

- Case 3 – Predicting the future state: The aim, in this case, is to predict the future state of the PT to optimize the performance in the PE. Define $T^f_m$ as the prediction window from the present time $t$, the future state $t + T^f_m$ can be expressed as

$$
\tilde{Y}_m(\tau + (t - \tau + T^f_m)) = X_m(t + T^f_m)
$$

$$
= Y_m(t + T^f_m + D^c_m), \forall t, T^p_m = t - \tau + T^f_m, \quad (4)
$$

$$
T^p_m \leq T^p_{\text{max}}, D^c_m < T^p_m,
$$

where the prediction for the future time $t + T^f_m$ equals the actual data generated by the PT at time $t + T^f_m$ which is equivalent to the actual data received at time $t + T^f_m + D^c_m$. In this case,

$$
D^v_m = 0. \quad (5)
$$

### 3.2 PT-VT synchronization latency requirements

Predictions, however, come with huge challenges and the need to guarantee the required reliability. Thus, we introduced a verification system where every prediction is verified using its corresponding actual states. Generally, any actual generated state $X_m(t)$ should be received by the VT within a period $D_{req}$. However, the parameter $D^c_m$ is a random variable which can be obtained as

$$
D^c_m = D^q_m + D^s_m + T^f_m + D^dec_m, \quad (6)
$$

where $D^q_m, D^s_m, T^f_m$ and $D^dec_m$ represent the delays due to queuing, secured FML, model transmission and decoding, respectively. From [5], we can approximate $D^dec_m = \kappa D^m_m, \forall \kappa > 0$. With this, any $X_m(t)$ is expected to be received at the VE for prediction verification before the time slot $D_{req}$ with probability $1 - \epsilon^v_m$, where $\epsilon^v_m < \epsilon_{req}$ is the violation probability while $\epsilon_{req}$ is the maximum tolerable error probability. Thus, the prediction verification times for the three cases are $\mathcal{V}^{(1)} = D^c_m$, $\mathcal{V}^{(2)} = D^c_m - D^th_m$ and $\mathcal{V}^{(3)} = T^f_m + D^c_m$, respectively, with $\mathcal{V}^{(1)}, \mathcal{V}^{(2)}, \mathcal{V}^{(3)} \leq \mathcal{V}_{req}$, where $\mathcal{V}_{req}$ is the acceptable verification time threshold. Clearly, $\mathcal{V}^{(3)}$ is directly proportional to $T^f_m$ while $\mathcal{V}^{(1)}$, $\mathcal{V}^{(2)}$ and $\mathcal{V}^{(3)}$ are all proportional to $D^c_m$.

### 3.3 Reliability requirement

From (6), we know that the overall reliability of the proposed PeHDT depends on the maximum PT-VT synchronization delay violation probability as well as the prediction error probability – defined as the probability that the prediction error is greater than the predefined threshold called the just noticeable difference (JND). Let the difference...
between $\bar{y}_m(t)$ and $Y_m(t + D_{m}^p)$ be represented as $e^{(1)}_m(t) = [e^{(1)}_m(t)]_{i = 1, \ldots, f}^T$, where $e^{(1)}_m(t) = \bar{y}_m(t) - y_m(t + D_{m}^c)$. We can obtain the prediction error for case 1 via the mean square error (MSE) such that

$$E^{(1)}_m(t) = \text{MSE}(\bar{y}_m(t) - Y_m(t + D_{m}^p)) = \frac{1}{f} \sum_{i=1}^{f} (\bar{y}_m^i(t) - y_m^i(t + D_{m}^c))^2.$$  

(7)

Similarly for cases 2 and 3,

$$E^{(2)}_m(t) = \text{MSE}(\bar{y}_m(t - D_{m}^{th}) - Y_m(t + D_{m}^c - D_{m}^{th})),$$  

(8)

$$E^{(3)}_m(t) = \text{MSE}(\bar{y}_m(t + T_{m}^p) - Y_m(t + T_{m}^p + D_{m}^p)).$$  

(9)

We define the JND of this system as $J_{ND} = [J_{ND}]_{i = 1, \ldots, f}^T$. Hence, the prediction error probability is given as

$$e^{(s)}_m(t) = 1 - \prod_{i=1}^{f} \Pr(|e^{(s)}_m(t)| \leq J_{ND}), \forall s \in \{1, 2, 3\}.$$  

(10)

The overall reliability of such a system can be defined as

$$e^{(m)}_m = 1 - (1 - e^{(s)}_m)(1 - e^{(s)}_m).$$  

(11)

Generally, since $e^{(s)}_m$ and $e^{(m)}_m$ are small, the overall reliability $e^{(m)}_m \approx e^{(s)}_m + e^{(s)}_m \leq \varepsilon_{req}$, where $\varepsilon_{req}$ captures the reliability requirement. The parameter $e^{(m)}_m$ will be obtained in the next section.

### 4 Performance Analysis

This section analyzes the proposed PeHDT scheme by characterizing the prediction process as well as the PT-VT synchronization delay. This will be essential for the problem formulation and optimization presented in the next section.

#### 4.1 Prediction process

To characterize the prediction process, we need to obtain the relationship between the prediction horizon and the prediction error. For explanation purposes, we consider a general linear prediction approach as in [5], [14], [18], where the transition from any state $t$ to state $t + 1$ of any typical PT $X_m$ follows the state transition function [25] as

$$X_m(t + 1) = \Phi_m X_m(t) + W_m(t),$$  

(12)

where $\Phi_m = [\varphi_{i,j}]_{f \times f}$, $i, j = 1, 2, \ldots, f$, assumed to be constant, represents the state transition matrix while $W_m(t) = [w_m(t)]_{f \times 1}$, $i = 1, 2, \ldots, f$ depicts the transition noise and are independent random variables that are generated following the Gaussian distributions of mean zero and variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_f^2$. From (12), the expected state in the slot $t + T_{m}^p$ can be expressed as

$$X_m(t + T_{m}^p) = (\Phi_m)^{T_{m}^p} X_m(t) + \sum_{i=1}^{T_{m}^p} (\Phi_m)^{T_{m}^p - i} W_m(t + i - 1).$$  

(13)

Thus, given the state of the VT at any time slot $t$, we can predict its state in time slot $t + 1$ and $t + T_{m}^p$, respectively following the Kalman filter as

$$\bar{Y}_m(t + 1) = \Phi_m Y_m(t),$$  

$$\bar{Y}_m(t + T_{m}^p) = (\Phi_m)^{T_{m}^p} Y_m(t).$$  

(14)

With a slight abuse of notation, let $Y_m(t + T_{m}^p)$ and $\bar{Y}_m(t + T_{m}^p)$ represent the received and predicted states, respectively for any time slot $t + T_{m}^p$. The disparity between any $Y_m(t + T_{m}^p)$ and $\bar{Y}_m(t + T_{m}^p)$ can be expressed as

$$E_m(t + T_{m}^p) = Y_m(t + T_{m}^p) - \bar{Y}_m(t + T_{m}^p) = W_m(t + T_{m}^p - 1) + \sum_{i=1}^{T_{m}^p - 1} (\Phi_m)^{T_{m}^p - 1} W_m(t + i - 1).$$  

(15)

The jth element of $E_m(t + T_{m}^p)$ is obtained as

$$e^{(s)}_m(t + T_{m}^p) = w_m(t + T_{m}^p - 1) + \sum_{i=1}^{T_{m}^p - 1} \sum_{n=1}^{f} \varphi_{m,j,n,T_{m}^p - i} w_m(t + i - 1),$$  

(16)

where $\varphi_{m,j,n,T_{m}^p - i}$ is the element of $(\Phi_m)^{T_{m}^p - 1}$ at the jth row and the n'th column. Since the considered state transition noises follow independent Gaussian distributions, $e^{(s)}_m(t + T_{m}^p)$ also follows Gaussian distribution with mean zero and variance $\rho_{m,j}(T_{m}^p)$ which equals to

$$\rho_{m,j}^2(T_{m}^p) = \sigma_j^2 + \sum_{i=1}^{j} \sum_{n=1}^{f} \left(\varphi_{m,j,n,T_{m}^p - i}\right)^2 \sigma_n^2.$$  

(17)

From (10),

$$\Pr(|e^{(s)}_m(t + T_{m}^p)| \leq J_{ND}) = 1 - \Pr(|e^{(s)}_m(t + T_{m}^p)| > J_{ND}) = 1 - \Psi_{T_{m,j}}(J_{ND})$$  

(18)

$$= 1 - \Psi\left(-\frac{J_{ND}}{\rho_{m,j}(T_{m}^p)}\right),$$

where $\Psi_{T_{m,j}}(.)$ is the cumulative distribution function (CDF) of $e^{(s)}_m(t + T_{m}^p)$ while $\Psi(.)$ represents the CDF of the standard Gaussian distribution with mean and variance of zero and one, respectively. From (10) and (18),

$$e^{(m)}_m = 1 - \prod_{j=1}^{f} \left[1 - \Psi\left(-\frac{J_{ND}}{\rho_{m,j}(T_{m}^p)}\right)\right].$$  

(19)

As proved in [5], $e^{(m)}_m$ strictly increases with $T_{m}^p$.

#### 4.2 PT-VT Synchronization delay

To obtain analysis for the PT-VT synchronization delay $D_{m}^p$, we characterize a typical VT updating process as a single server queueing system, where both the queuing and the service times are taken into consideration when studying the average delay between any PT-VT pair. The queuing time $D_{m}^{q}$ captures the average time each batch waits before processing and includes the wait-to-batch time. We consider the batch processing to follow the first come first serve (FCFS) approach, where the first formed batch proceeds to processing immediately after the server is idle subject to energy availability while the other batches remain in the queue. Similarly, the service time captures the processing time of any batch and includes the secured FL processing time, transmission time and decoding time.

According to [26], [27], the VT updating process of any LA $m$ can be modelled as a finite-buffer batch service queue system with batch-size dependent service. In such a system, the overall PT-VT synchronization delay depends on the
size of the batch $B_{\text{min}} \leq B_m \leq B_{\text{max}}$ and $Q$. Hence, the service time of any batch under processing depends on the number of packets it contains and is known to be arbitrarily distributed. In addition, we consider the packets inter-generation time of any LA $m$ to be independent and geometrically distributed with probability mass function (PMF) $a_k^m = P(A = k) = \lambda_m k - 1 \lambda_m$, $0 < \lambda_m < 1, k \geq 1$, such that arrivals (i.e., generations) can only occur in the interval $(t_-, t)$ while departures (i.e., services) can only occur in the interval $(t, t_+$. This packet generation time follows the Bernoulli process and packets are batched for service following the FCFS approach.

Let the number of packets waiting in the queue upon completing the $u$th service be represented as $Q_{u}^+$, while the number of newly generated packets throughout the processing time of the $u$th + 1 service with batch size $B_{u+1}$ is denoted as $A_{u+1}$. The queue distribution of the proposed PeHDT scheme can be expressed as

$$Q_{u+1}^+ = \min\{(Q_u^+ - B_{u+1})^+, A_{u+1}, Q\},$$

where $(x)^+ = \max(x, 0)$. We know that the probability distribution of $A_{u+1}$ can be obtained as

$$d_k^B = P(A_{u+1} = k) = \sum_{n=1}^\infty P(A_{u+1} = k | B_{u+1} = n) \times$$

$$P(B_{u+1} = n) = \sum_{n=k}^\infty \frac{n}{k} \lambda_m^{n-k} \left(1 - \lambda_m\right)^{n-k} s_n, k \geq 0,$$

where $s_n$ is the PMF of the independent and identically distributed batch processing time. Define $p_{n,B}^+, \forall 0 \leq n \leq Q, B_{\text{min}} \leq B_m \leq B_{\text{max}}$, as the probability of having $n$ packets in the queue immediately after the successful processing of a batch of size $B_m$. We can obtain $p_{n,B}^+$ by solving the steady-state equations $\pi P = \pi$ and $\pi 1 = 1$ [27], where $\pi = (\pi_0^+, \pi_1^+, ..., \pi_Q^+)$, such that $\pi_n^+$ is a row vector of dimension $d = (B_{\text{max}} - B_{\text{min}} + 1)$, and the transition probability $P$ can be obtained as

$$p_{n,B}^+ = \begin{pmatrix} 0 & 1 & \ldots & n & Q - 1 & Q \\ A_{0,1} & A_{1,1} & \ldots & A_{n,1} & A_{Q-1,1} & B_{Q,1} \\ 1 & A_{0,1} & A_{1,1} & \ldots & A_{n,1} & A_{Q-1,1} & B_{Q,1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

where $A_{n,k}$ and $B_{n,k}$ have a dimension of $d$ and can be expressed as

$$A_{n,k} = I_d^T \otimes \mathbf{K}, 1 \leq k \leq d, 0 \leq n \leq Q - 1,$$

$$B_{Q,k} = \mathbf{1}_k^T \otimes \mathbf{K} + I_d^T \otimes \mathbf{K}, 1 \leq k \leq B_{\text{max}} - B_{\text{min}},$$

$$B_{n,d} = \mathbf{1}_n^T \otimes \mathbf{K}, B_{\text{max}} \leq n \leq Q,$$

where $\mathbf{1}_k$ is a column vector of dimension $d$ with all elements equal to $1$ except at the $k$th position where it equals $1$, $\mathbf{K}$ is a column vector of dimension $d$ consisting of $d_k^B$, while $\mathbf{K}$ is a column vector of dimension $d$ consisting of

$$(1 - \sum_{n=0}^Q d_k^B)$$

and the operator $\otimes$ represents the Kronecker product of two matrices.

At the beginning of any time slot and before a potential arrival at $t_-$, let $Q_q^{(t_-)}$ be the number of packets waiting in the queue, $B_l^{(t_-)}$ be the number of packets in the batch currently under processing if there is one, and $U_l^{(t_-)}$ be the remaining service time of such a batch. Then

$$P_{n,B}^{(t_-)} = \begin{cases} P_{n,0}^{Q_q^{(t_-)}} = n, B_l^{(t_-)} = 0, \\ \forall 0 \leq n \leq B_{\text{min}} - 1 \\ P_{n,B,l}^{Q_q^{(t_-)}} = n, B_l^{(t_-)} = B, U_l^{(t_-)} = l, \\ \forall l > 0, 0 \leq n \leq Q, B_{\text{min}} \leq B \leq B_{\text{max}}. \end{cases}$$

(23)

From (23), we can define $p_{n,0} = \lim_{t_- \to \infty} P_{n,0,l}^{(t_-)}$ and $p_{n,B} = \lim_{t_- \to \infty} P_{n,B,l}^{(t_-)}$. By further considering the states of the system at $(t_-)$ and $(t_+ + 1)$ in the steady-state, we can obtain $p_{n,0}$ and $p_{n,B}$ for $l \geq 1$ by following the probabilistic argument. Since the probability of having $n$ packets in the queue at any departure-epoch $p_n^+ = \sum_{B = B_{\text{min}}}^{B_{\text{max}}} p_{n,B}^t$, $\forall 0 \leq n \leq Q$, we have

$$p_{n,0} = E^{-1} \left( \sum_{j=0}^n p_j^+ \right), \forall 0 \leq n \leq B_{\text{min}} - 1,$$

(24)

$$p_{n,B} = E^{-1} \left( s_n p_n^+ - \sum_{j=0}^n p_j^+, B \right),$$

$$\forall B_{\text{min}} + 1 \leq B \leq B_{\text{max}} - 1, 0 \leq n \leq Q - 1,$$

where

$$E = \lambda_m \left( s_{B_{\text{min}}} \sum_{n=0}^{B_{\text{min}}-1} p_n^+ + \sum_{n=B_{\text{min}}+1}^{B_{\text{max}}} s_n p_n^+ + s_{B_{\text{max}}} \sum_{i=B_{\text{max}}+1}^Q (B_{\text{min}} - i) p_i^+ \right).$$

(25)

The average queueing time of each packet at LA $m$ can thus be obtained as

$$D_m^q = \frac{\sum_{n=0}^Q n P_n^q}{\lambda_m (1 - \sum_{n=0}^{B_{\text{max}}} \sum_{B_{\text{min}}} P_{n,B} Q_{n,B} - B_{\text{min}}) - 2 B_{\text{min}}},$$

(26)

where $P_n^q$ captures the distribution of the number of packets in the queue of PT $m$, which can be obtained as

$$P_n^q = \begin{cases} p_{n,0} + \sum_{B_{\text{min}}}^{B_{\text{max}}} p_{n,B}^m p_{n,B_m} \leq 0 \leq B_{\text{min}} - 1, B_{\text{min}} \leq n \leq Q. \end{cases}$$

(27)

Generally, we know that $D_m^q$ can be obtained by dividing the average number of packets in the queue by the effective arrival rate. This effective arrival rate depends on the probability of blocking – the probability that there is an arrival when the service has reached its full queuing capacity $Q + B_{\text{max}}$. From this, (27) can be derived.

Next, we obtain the average service time of any typical packet. To do this, we need to find the average time to complete the secured FML and transmission processes.
4.2.1 Secured FML process delay

To obtain the total time required to carry out a secured FML process for each update, we considered a typical FML process where the aim is to produce a task-specific model \( \omega^{ts} \) that represents the actual states of the tagged PT \( m \). To achieve this, each of the \( M \) selected neighbouring LAs participates in the training of the global model \( \omega_{GA} \) to improve the accuracy of the task-specific model \( \omega^{ts} \) through parameters sharing. Each LA, after every round of training, transmits its trained local model to the GA for the global model aggregation. The total time of local training over \( J_m - 1 \) communication rounds given that each LA \( i \) requires \( f_i \) number of CPUs to execute one sample of training data is given as

\[
T_{m}^{exe} = \max_{i \in [M]} \left( \sum_{j=1}^{J_m-1} f_i |S_i(j)| / f_i \right),
\]

where \( f_i \) is the CPU frequency of any LA \( i \) and \( |S_i(j)| \) is the size of its locally batched data. \( T_{m}^{exe} \) is calculated over \( J_m - 1 \) rounds since the last round is used for validation. For the model aggregation at the GA, the time required to perform the global aggregation over \( J_m \) communication round is given as

\[
T_{GA}^{exe} = \sum_{j=1}^{J_m} \left( f_{agg} \sum_{i=1}^{M} |\omega_i^{sh}(j)| / f_{GA} \right),
\]

where \( f_{GA} \gg f_m \) is the CPU frequency of GA, \( |\omega_i^{sh}| \) is the size of \( \omega_i^{sh} \) and \( f_{agg} \) is the number of CPU cycles required to aggregate one trained local model. The transmission time from LA to GA \( T_{LA-GA} \) and from GA to LA \( T_{GA-LA} \) during the training process can be respectively obtained as

\[
T_{LA-GA} = \sum_{j=1}^{J_m} \max_{i \in [M]} \left( \frac{|\omega_i^{sh}(j)|}{r_i(t)} \right),
\]

\[
T_{GA-LA} = \sum_{j=1}^{J_m} \frac{|\omega_{GA}(j)|}{r_{GA}},
\]

where \( |\omega_{GA}| \), \( r_i \) and \( r_{GA} \) represent the size of the global model, the achievable data rate of LA \( i \) and the achievable data rate of the GA, respectively.

For any typical PT-VT pair, the average time required to carry out a secured FML process over \( J_m \) communication rounds for any typical packet is equivalent to

\[
D_{FL} = \frac{1}{B_m} \left( \sum_{j=1}^{J_m} \left( f_{agg} \sum_{i=1}^{M} |\omega_i^{sh}(j)| / f_{GA} \right) + \max_{i \in [M]} \left( \frac{|\omega_i^{sh}(j)|}{r_i(t)} \right) \right)
\]

\[+ \frac{|\omega_{GA}(j)|}{r_{GA}} \left( + \max_{i \in [M]} \left( \frac{|\omega_m(J_m)|}{v_i} + \sum_{j=1}^{J_m-1} f_i |S_i(j)| / f_i \right) \right),
\]

where \( v_i \) is the validation capacity of each node. The total delay caused as a result of the adopted secured FML technique includes the time required for local training, model transmissions, global aggregation and blockchain-enabled validation which is carried out during the final communication round of the FL process. Hence, (31) follows from (28)–(30).

4.2.2 Offloading delay and energy consumption analysis

After the successful completion of a secured FL process, the tagged PT through its LA \( m \) will offload its trained task-specific model to its paired VT. Since such offloads are important for verifying the previous predictions, the offloads should be received with no or a limited number of retransmissions. Following the assumption of homogeneous PPs, the path-loss model is considered with the signal power decaying at a rate \( r^{-\alpha} \), where \( r \) and \( \alpha \) represent the propagation distance and path-loss exponent, respectively. We also considered the Rayleigh fading multipath environment where the actual channel power gain \( h^{AC} \) and the interfering channel power gain \( h^{in} \) are exponentially distributed with unit mean. A full path-loss inversion power control \( \rho_m \) is used by each LA.

By adopting the power ramping scheme, each typical PT LA selects a suitable offloading power \( \rho_m \in [\rho_1, \rho_2, ..., \rho_B] \), \( \forall \rho_1 < \rho_2 < ... < \rho_B \) for offloading based on its energy capacity to prolong its lifetime by saving energy. The power \( \rho_B \) is the offloading power threshold and \( B \) is the maximum index of offloading. In the event of an unsuccessful offloading with power \( \rho_l \), the LA selects power \( \rho_{l+1} \), \( \forall \rho_{l+1} < \rho_B \) for retransmission to improve the chances of successful offloading. For any \( l \)th offloading attempt, the probability that the trained model offloaded by any tagged LA \( m_0 \in \Phi_{PT} \) is successfully received by its corresponding VT \( V_0 \in \Phi_{VT} \) can be obtained as

\[
Pr_l^{suc} = P \left( \frac{\rho_l h_0^{AC}}{\sigma^2 + I_{in} + I_{out}} > \theta \right) = \exp \left( - \frac{\theta \sigma^2}{\rho_l} \right) \prod_{b=1}^{B} L_{in}^{(b)} \left( \frac{\theta}{\rho_l} \right) L_{out}^{(b)} \left( \frac{\theta}{\rho_l} \right),
\]

where \( h_0^{AC} \) is the channel gain between the LA \( m_0 \) and its corresponding VT \( V_0 \), \( \sigma^2 \) is the noise signal power, \( I_{in} \) captures the intra-cell interference, \( I_{out} \) depicts the inter-cell interference, while \( \theta \) is the successful transmission threshold. The parameters \( L_{in}^{(b)} \) and \( L_{out}^{(b)} \) represent the Laplace transforms of the interference \( I_{in} \) and \( I_{out} \), respectively received at \( V_0 \), and were obtained following the probability generating function of the PPP. At \( \alpha = 4 \), we can obtain \( Pr_l^{suc} \) in a closed-form as

\[
Pr_l^{suc} = \exp \left( - \frac{\theta \sigma^2}{\rho_l} \right) \prod_{b=1}^{B} \exp \left\{ - \frac{\theta \Delta_b}{\rho_l} \right\} \frac{\theta \Delta_b}{\rho_l} \left[ 1 + \frac{\theta \Delta_b}{(1 + \theta \Delta_b) 3.575} \right],
\]

where \( \Delta_b = [\Delta_1, \Delta_2, ..., \Delta_B] \) represents the marginal distribution of transmission power phases. The proof of (33) follows from [22], [28], [29] and is omitted to avoid unnecessary repetition. From this, we can obtain the average offloading success probability following the law of total probability and conditioned on the tagged LA \( m_0 \in \Phi_{PT} \) as

\[
Pr_{off}^{suc} = E[Pr_l^{suc}] = \sum_{l=1}^{B} \Delta_l Pr_l^{suc},
\]
where $\mathbb{E}[X]$ is the expected value of $[X]$. Hence, the packet transmission delay of PT $m$ is

$$D_m^t = \frac{1}{\mathcal{B}_m \mathbb{E}[P_l^m]} = \frac{1}{\mathcal{B}_m (\sum \mathcal{A}_l (P_l^m))}. \quad (35)$$

Define the service state of LA $m$ at time slot $t$ as $\emptyset(t)$, such that $\emptyset(t) = 1$ when any LA $m$ is in the service mode and $\emptyset(t) = 0$ when waiting either due to batch formation and energy harvesting. We assumed that each LA harvests energy from the activities of neighbouring LAs including the human body environment to power itself and the associated sensing devices. We know that the energy consumption in the system can be categorized into three components: sensing energy, queue monitoring energy and processing energy. Let $\xi_s$, $\xi_q$ and $\xi_t$ represent the sensing, queue monitoring and batch processing power consumptions, respectively. Then the total energy consumption of any LA $m$ in any time slot $t$ can be represented as

$$e_m^{ene}(t) = \xi_s \sum_{n=1}^{N} \lambda_n(t) + \xi_q \lambda_m(t) + \xi_t \mu_{batch}(m)(\emptyset(t)), \quad (36)$$

where $\lambda_n$ is the sensing rate of device $n$ and $\mu_{batch,m} = \frac{1}{\mathcal{B}_m}$ is the batch processing rate of LA $m$. Note that $\lambda_m(t)$ in (36) is bounded by $\lambda_{max}$ and $\mu_{batch,m}(t)$ is bounded by the maximum batch rate $\mu_{batch,m}$ with maximum $w_m(t) = 1$. Thus, the maximum total energy consumption can be obtained as

$$e_m^{ene} = (\xi_s + \xi_q) \lambda_{max} + \xi_t \mu_{min_{batch,m}}. \quad (37)$$

Given that the length of the energy queue for any LA $m$ is $E_m(t)$, we can define the time-varying process of $E_m(t)$ following

$$E_m(t+1) = E_m(t) - e_m^{ene}(t) + e_m^{in}(t), \quad (38)$$

where $e_m^{in}(t)$ is the energy input process. Intuitively, the battery capacity $C_m \geq E_m, \forall m \in [M].$

### 4.2.3 Violation probability

Based on the obtained delay analysis, we can study the violation probability at any given instance. Denote $D_m^h$, as the time duration during which the synchronization delay $D_m^h(t) > D_{req}$ between any time slot $T_k^{h-1}$ and $T_k^h$. The violation probability reflects the reliability parameter $\epsilon_m^v$ and captures the probability that the synchronization delay $D_m^h$ violates the delay requirement threshold. This violation probability can be expressed as

$$\epsilon_m^v = P\{D_m^h > D_{req}\} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} I(D_m^h(t) > D_{req})$$

$$= \lim_{T \to \infty} \frac{1}{T} \sum_{h=1}^{H(T)} D_m^h = \lim_{T \to \infty} \frac{H(T)}{T} \sum_{h=1}^{H(T)} \frac{D_m^h}{H(T)} \quad (39)$$

$$= \lim_{T \to \infty} \frac{H(T)}{T} \sum_{h=1}^{H(T)} \frac{D_m^h}{H(T)} \sum_{l=1}^{H(T)} (T_k^l - T_k^{l-1}) = \mathbb{E}[D_m^h] = \mathbb{E}[T_k - T_k^{l-1}],$$

where $H(T)$ is the number of successful VT updating processes until the time $T$ and $I(a)$ is the indicator function that is 1 when condition $a$ is true and 0 otherwise. By taking the overall delay threshold $D_{req} \geq 1$, we can obtain the average violation probability as

$$\epsilon_m^v = \left(1 - \frac{1}{D_m}\right)^{|D_{req}|}, \quad (40)$$

where $\lfloor . \rfloor$ is the floor function. From (40), the relationship between $D_m^h$ and $\epsilon_m^v$ can be successfully obtained.

## 5 Problem formulation and optimization

The batching-enhanced strategy reduces the communication cost through group processing although may also compromise the delay constraints. There is, therefore, a need to derive the relationship between the worst-case PT-VT synchronization delay and maximum batch size. As a result, we aim to optimize the overall reliability of the proposed PeHDT system while minimizing the communication cost by first determining an optimal system setting of batching and prediction horizon for any typical PT-VT pair. To do this, we formulate the optimization problem as follows;

$$\min_{\lambda_m, B_m, T_m} \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} \epsilon_m^{ene}$$

s.t. $\epsilon_m^{v} + \epsilon_m^{p} \leq \epsilon_{req}$, \quad (41a)

$\gamma_m^{(1)}, \gamma_m^{(2)}, \gamma_m^{(3)} \leq \gamma_{req}$, \quad (41b)

$B_{min} \leq B_m \leq B_{max}$, \quad (41c)

$e_m^{ene} \leq E_m \leq C_m$, \quad (41d)

$T_m^{p} \leq T_{max}^{p}$, \quad (41e)

$\sum_{n} \lambda_n(t) \leq \lambda_{max}(t) \leq \lambda_{max}$, \quad (41f)

$\rho_1 < \rho_2 < \ldots < \rho_B$. \quad (41g)

The constraint in (41a) ensures that the overall error probability for each LA $m \in [M]$ is less than the acceptable error probability, while (41b) shows that the prediction verification time must be below the maximum acceptable threshold. In addition, (41c) ensures that the batch size selected by LA $m$ is within acceptable thresholds necessary to reduce the communication cost, which is generally a function of both time and energy costs [2]. Constraint (41d) indicates the energy requirement while (41e) ensures the limit of the prediction horizon. Finally, (41f) and (41g) ensure the arrival rate at any LA $m$ and the offloading power threshold, respectively fall within the acceptable limits, given that $\lambda_n$ is the data sampling rate of each sensing device associated to LA $m$.

### 5.1 Constrained deep reinforcement learning

Directly solving (41) is difficult, if not impossible, because of the stochastic nature of parameters $T_m^{p}$ and $B_m, \forall m \in [M]$. To address this complexity, we re-formulate (41) as a cMDP problem. The formulated cMDP problem is later solved by applying the deep reinforcement learning (DRL) algorithm. This ensures we obtain a suitable weighting coefficient, through optimization of the Lagrangian multiplier, capable of satisfying the reliability constraint. To achieve this, we incorporated the primary-dual DDPG algorithm and demonstrated its ability to provide a solution to the
optimization problem. Generally, the constrained MDP can be formulated by first defining its state space, action space, reward function, constraint function and policy. The state at any decision epoch can be given as
\[ s^{(t)} = (\lambda_m(t), E_m(t), T_p^m(t)), \] (42)
where \( \lambda_m \) is the arrival rate of any LA \( m \). Similarly, the action space at any decision epoch can be represented as
\[ a^{(t)} = (B_m(t), \rho_l(t), r_m(t), r_{GA}(t)). \] (43)

Given the state and the action at the end of time \( t \), the instantaneous reward \( r^{(t)} \) can be expressed as a function of the batch processing rate \( \mu_{batch,m} \) and the prediction horizon \( T_p \) while the cost \( c^{(t)} \) can be expressed as the reliability parameter \( \epsilon_m \), such that
\[ r^{(t)} = \mu_{batch,m}(t + 1) + \frac{T_p^m(t + 1)}{T_p^\text{max}}, \] (44)
\[ c^{(t)} = \epsilon_m(t + 1). \] (45)

In such a system, we consider the agent to follow a deterministic policy with parameter \( \vartheta \) given as \( \mu : a^{(t)} = \mu(s^{(t)}|\vartheta) \), where the long-term discounted reward is obtained as
\[ R_\mu(\cdot|\vartheta) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r^{(t)} \right], \] (46)
where \( \gamma \in [0,1) \) is generally known as the discount factor. Under the policy \( (\cdot|\vartheta) \), the long-term discounted cost can as well be obtained as
\[ C_\mu(\cdot|\vartheta) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t c^{(t)} \right]. \] (47)

Generally, the cMDP aims to find the optimal policy \( \mu^*(\cdot|\vartheta^*) \) that maximizes \( R_\mu(\cdot|\vartheta) \) subject to \( C_\mu(\cdot|\vartheta) \). Hence,
\[ \mu^*(\cdot|\vartheta^*) = \arg \max_{\mu(\cdot|\vartheta)} R_\mu(\cdot|\vartheta) \] s.t. \( C_\mu(\cdot|\vartheta) \leq \frac{\epsilon_{\text{req}}}{1 - \gamma} \). (48)

Following [30], [31] the Lagrangian function of (48) can be formulated as
\[ \epsilon(\mu(\cdot|\vartheta), \lambda) = R_\mu(\cdot|\vartheta) - \lambda \left( C_\mu(\cdot|\vartheta) - \frac{\epsilon_{\text{req}}}{1 - \gamma} \right), \] (49)
where \( \lambda \) is the Lagrangian multiplier. We can then convert the constrained problem to an unconstrained problem
\[ (\mu^*(\cdot|\vartheta^*), \lambda^*) = \arg \min_{\lambda \geq 0} \epsilon(\mu(\cdot|\vartheta), \lambda). \] (50)

5.2 Primary-dual DDPG

Because of the existence of constraints in the formulated cMDP problem, we adopted a primary-dual DDPG algorithm with a primal-dual approach to simultaneously discover the optimal policy and the corresponding dual variable. We define the policy, the long-term reward and the long-term cost using neural networks as \( \mu(\cdot|\vartheta), Q^R(\cdot|\vartheta^R) \) and \( Q^C(\cdot|\vartheta^C) \), respectively, where \( \vartheta, \vartheta^R \) and \( \vartheta^C \) represent the parameters of these neural networks. The critic and cost networks can then be optimized using the Bellman equations such that the target reward and cost are respectively given as
\[ Q^R(s^{(t)}_i, a^{(t)}_i) = r + \gamma Q^R(s^{(t+1)}_i, \mu(s^{(t+1)}_i|\vartheta)|\vartheta^R), \] (51)
\[ Q^C(s^{(t)}_i, a^{(t)}_i) = r + \gamma Q^C(s^{(t+1)}_i, \mu(s^{(t+1)}_i|\vartheta)|\vartheta^C), \]
where \( i = 1, 2, ..., N_{\text{b}} \) and \( N_{\text{b}} \) is the learning batch size. From this, the mean-squared Bellman error loss function can be adopted to update the neural networks of both the long-term reward and long-term cost as
\[ L(\vartheta^R) = \frac{\sum_{i=1}^{N_{\text{b}}} \left( (Q^R(s^{(t)}_i, a^{(t)}_i) - Q^R(s^{(t)}_i, a^{(t)}_i|\vartheta^R))^2 \right)}{N_{\text{b}}}, \] (52)
\[ L(\vartheta^C) = \frac{\sum_{i=1}^{N_{\text{b}}} \left( (Q^C(s^{(t)}_i, a^{(t)}_i) - Q^C(s^{(t)}_i, a^{(t)}_i|\vartheta^C))^2 \right)}{N_{\text{b}}}, \]
By maximizing (49), the actor policy of the primary-dual DDPG is updated by replacing the long-term reward with the critic network and the long-term cost with the cost network such that
\[ \max_{\vartheta} \mathbb{E}[Q^R(s^{(t)}_i, a^{(t)}_i) - \lambda Q^C(s^{(t)}_i, a^{(t)}_i)]. \] (53)
We can then update \( \lambda \) using the gradient descent to minimize (49) based on
\[ \lambda^{(t+1)} = \lambda^{(t)} + \beta_\epsilon \left( Q^C(s^{(t)}_i, \mu(s^{(t)}_i|\vartheta)|\vartheta^C) - \frac{\epsilon_{\text{req}}}{1 - \gamma} \right), \] (54)
where \( \beta_\epsilon \) captures the step size. Given that \( \zeta \) is the smoothing factor, the algorithm for the primary-dual DDPG is presented as Algorithm 1.

6 Numerical simulations

In this section, we evaluate the performance of the proposed PeHDT framework and demonstrate the ability of the adopted primary-dual DDPG algorithm in better capturing the PeHDT framework than the traditional DDPG algorithm. We also compare the performance of our proposed communication, batching and prediction verification solution under different cases, as discussed in Section 3.6.1 Simulation environment

We implemented the proposed PeHDT framework using PyCharm – an integrated development environment for programming and software development primarily focused on the Python programming language. To solve the problem in (41), we incorporated the primary-dual DDPG algorithm into the PeHDT framework. The performance of the proposed framework was then evaluated under different scenarios: Predicting the present state (Case 1), wait-before-predicting the latest state (Case 2), and predicting the future state (Case 3). We also simulated the existing approach with no batching as a performance benchmark, by letting \( B_m = 1 \) under an optimal prediction horizon. For case 1, the system always predicts the present state using the last received actual data while for the case 2, the system predicts the present state after waiting for a predetermined period \( D_m^{th} \geq 0 \). We later adopted the conventional DDPG...
Algorithm 1: Primary-dual DDPG

Initialize the reward critic, cost critic and actor networks \( Q^R(s_i(t), a_i^R(t) | \theta^R) \), \( Q^C(s_i(t), a_i^C(t) | \theta^C) \), \( \mu(s_i(t) | \theta) \).

Initialize the target networks \( \theta_{\text{targ}} \leftarrow \theta \), \( \theta_{\text{targ},1} \leftarrow \theta^R_1 \), \( \theta_{\text{targ},2} \leftarrow \theta^C_2 \).

Initialize empty replay buffer and the dual variable \( R, \mathcal{R} \).

For episode \( k = 0, 1, \ldots \), do

Initialize a random process \( N_{\text{ran}} \) for action exploration

Receive initial state \( s_0 \sim p_0 \)

For \( t = 1, \ldots, T \) do

Select \( a_i(t) = \mu(s(t) | \theta) + N_{\text{ran}} \)

Execute \( a_i(t) \) and observe the reward, cost, the next state and store \( (s_i(t), a_{i}^R(t), r^R_i(t), c_i^C(t), s_i(t+1)) \) in \( \mathcal{R} \)

Randomly sample a batch of \( N_b \) transitions,

\[ \{(s_i^t, a_{i}^R, r_i^R(t), c_i^C(t), s_i(t+1)) \}^N_{t=0} \]

Update \( Q^R \) and \( Q^C \) functions by one step of gradient ascent

Update the actor policy by one step of gradient ascent using (53)

Update the dual variable by one step of gradient ascent using (54)

Update the target networks:

\[ \begin{align*}
\theta_{\text{targ}}^R & \leftarrow \varsigma \theta_{\text{targ},1}^R + (1 - \varsigma) \theta^R_{1(t)}; \\
\theta_{\text{targ}}^C & \leftarrow \varsigma \theta_{\text{targ},2}^C + (1 - \varsigma) \theta^C_{2(t)}; \\
\theta_{\text{targ}} & \leftarrow \varsigma \theta_{\text{targ}} + (1 - \varsigma) \theta, \quad \forall \varsigma \in [0, 1], j = 1, 2.
\end{align*} \]

End For

End For

Table 2: Simulation parameters

<table>
<thead>
<tr>
<th>Notations</th>
<th>Values</th>
<th>Notations</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B_{\text{max}} )</td>
<td>10</td>
<td>( B )</td>
<td>10</td>
</tr>
<tr>
<td>( M )</td>
<td>20</td>
<td>( \xi_1 )</td>
<td>0.05 mJ/Kb</td>
</tr>
<tr>
<td>( T_{\text{max}} )</td>
<td>20 ms</td>
<td>( \xi_2 )</td>
<td>0.12 mJ/Kb</td>
</tr>
<tr>
<td>( Q )</td>
<td>25</td>
<td>( \xi_3 )</td>
<td>0.1 mJ/Kb</td>
</tr>
<tr>
<td>( D_{\text{req}} )</td>
<td>14 ms</td>
<td>( \sigma^2 )</td>
<td>-180 dBm</td>
</tr>
<tr>
<td>( \epsilon_{\text{req}} )</td>
<td>10^{-1}</td>
<td>( r_{GA}(t) )</td>
<td>( r_{GA} \in [0.6, 0.8] ) GHz</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>0.1</td>
<td>( v_{GA}, f_{GA} )</td>
<td>[15, 20] GHz</td>
</tr>
<tr>
<td>( \rho_B )</td>
<td>23 dBm</td>
<td>(</td>
<td>S_m(j)</td>
</tr>
<tr>
<td>( \lambda_{\text{max}} )</td>
<td>0.4</td>
<td>( f_{\text{agg}} )</td>
<td>1 GHz</td>
</tr>
<tr>
<td>( \theta )</td>
<td>0 dB</td>
<td>( r_m(t) )</td>
<td>( r_m \in [0.35, 0.4] ) GHz</td>
</tr>
<tr>
<td>( J_m )</td>
<td>20</td>
<td>( f_{GA} )</td>
<td>20 GHz</td>
</tr>
</tbody>
</table>

The performance evaluation

In Fig. 2, we investigate the performance of the proposed PeHDT framework using the average long-term discounted reward as the number of epochs increases. For cases 1, 2 and 3, the system searches for an optimal batch size \( B_m \) while \( B_m \) is always set to 1 for no batching scenario. Since case 3 relies on optimal values of both prediction horizon \( T_m \) and batch size \( B_m \), the performance of the proposed PeHDT framework is the best after convergence compared to case 1, case 2 and no batching scenario. The results in Fig. 2 also show that the conventional DDPG (also adopted to capture case 3) could achieve a better reward compared to its corresponding primary-dual DDPG. It is because the conventional DDPG is not limited by the cost constraint thus allowing the agent to pursue a higher reward. However, as shown in the next figure, DDPG could not meet the long-term cost constraint as defined in the optimization problem. Therefore, DDPG could be counted as the upper-bound of the performance that the proposed PeHDT framework could achieve.

Fig. 3 demonstrates the performances of different considered cases in terms of the average long-term discounted cost, which captures the cumulative cost that an agent incurs over time when interacting with an environment. As expected, case 3 enforced cost constraints successfully over a potentially higher time horizon with its ability to better balance exploration and exploitation to minimize long-term costs and maximize rewards over time. Interestingly, case 1 incurs the least cost, especially at the early stage (though also received a low reward). It is because the prediction horizon in such a case is always close to 1 resulting in a small prediction error.

The performance of the proposed communication, batching and prediction verification scheme in terms of the PT-VT synchronization delay is shown in Fig. 4, where case 3 again experiences the lowest latency while the highest delay is experienced in case 1. As \( B_m \) increases, the overall PT-VT synchronization delay continues to reduce until an optimal point for \( B_m \) and begins to rise after, for all cases. Since the existing prediction-enhanced scheme with no batching rule is equivalent to a scheme with \( B_m = 1 \), the expected delay is intuitively constant and is, therefore, omitted from the result. As observed in Fig. 4, the primary-dual DDPG-enhanced solution better captures the proposed PeHDT
In Fig. 5, we investigate the overall reliability (which relies on both the prediction error probability and the overall violation probability of the delay requirement) of the system as the time epoch increases. Interestingly, case 1 achieves the best overall reliability since the prediction horizon in such a case is always close to 1 resulting in a low prediction error which has a significant effect on its overall reliability. As the time epoch increases, case 3 achieves a reliability level closer to case 1 which demonstrates its ability to learn better from experience over time. In addition, case 1 achieves a relatively stable overall reliability because of the limited effect of prediction error on its overall system reliability while case 2 experiences a higher performance deterioration owing to its high tendency to violate the delay requirement because of its wait before prediction mechanism.

Furthermore, the relationship between the overall error probability and the overall delay requirement $D_{req}$ was investigated as presented in Fig. 6. Expectedly, the overall error probability decreases as $D_{req}$ increases since an increase in $D_{req}$ reduces the probability of violating the delay requirement. In addition, the higher the just noticeable difference $J_{ND}$, the lower the overall error probability while a lower $J_{ND}$ increases the sensitivity of the prediction verification system thereby increasing the overall error probability. For the sake of comparison, we also obtained the overall error probability for the DDPG algorithm while setting the $J_{ND} = 0.1\%$. Next, the relationship between the overall reliability and the violation probability $\epsilon_{vm}$ is presented in Fig. 7, where the overall reliability is shown to reduce as $\epsilon_{vm}$ increases because of the difficulty of meeting $D_{req}$ as $\epsilon_{vm}$ increases. Interestingly, improved performance is observed for the existing prediction-enhanced scheme with no batching rule compared to case 2 owing to its adoption of a single packet transmission system ($B_m = 1$) while case 3 continues to obtain a reliability level closer to case 1 to demonstrate its efficient learning process. A similar result is obtained in Fig. 8, where the effects of delay incurred due to the federated multi-task learning process $D_{m}^{FL}$, on the overall reliability of the proposed
PeHDT system, is presented. For all considered cases including the traditional DDPG-based solution, the overall reliability is observed to decrease as $D_{m}^{FL}$ increases owing to an increased probability of violating the overall delay requirement of the system. In addition, the existing prediction-enhanced scheme with no batching rule continues to experience a significant performance deterioration as $D_{m}^{FL}$ increases since an increase in $D_{m}^{FL}$ increases the processing time of each single packet batch which causes the queueing delay of waiting packets to increase. Fig. 8 further demonstrates the benefits of the batching approach to improve the overall performance in prediction solutions.

Finally, the performance of the proposed PeHDT framework is investigated using the prediction verification time as an appropriate metric in Fig. 9. The prediction verification time depicts the average time required to validate any predicted data by the VT and it depends on the time such a prediction was made and the time its corresponding actual data was received from the tagged PT. Generally, the verification time is reduced as $B_{m}$ increases, under all considered scenarios, until an optimal point for $B_{m}$ and begins to increase which demonstrates the impact of $B_{m}$ to ensure a reasonable prediction verification time subject to the verification time constraint $V_{req}$.

While the introduction of the batching rule increases the prediction verification time as $B_{m}$ increases beyond the optimal point, the presented results showed that the batching process can generally improve the overall performance of the proposed PeHDT system through the reduction of the communication cost without compromising the overall reliability of the system. Through the adopted primary-dual DDPG algorithm, the proposed system is able to determine the optimal $T_{p}^{m}$ and $B_{m}$ thereby obtaining the best PT-VT synchronization delay that optimizes VT predictions verification time.

7 Conclusion

In this paper, we investigate a communication, batching and prediction verification scheme where the corresponding VT of any typical PT is maintained in real-time or in advance using the predicted states or data of such a PT. Contrary to many existing prediction-based solutions, we deployed the predictors at the VTs to reduce the rate of packet loss in transmission. Since such a prediction-based connectivity scheme relies on available historical states of the counterpart PT as well as the VT learning experience while predictions are generally not error-free, the prediction solution can suffer from a high prediction error. As a result, we introduced a new prediction verification system where the actual generated data are used to validate the predictions with the aim of improving the learning experience of each VT while improving the generated VT model.

In addition, we incorporated an update batching process to reduce the communication burden in HDT and formulated the resulting problem as a cMDP. We later employed the tool of DRL to provide solutions to the formulated problem. Although the proposed batching-enabled PeHDT framework is demonstrated to be capable of enhancing the communication experience in HDT, the prediction solution has been designed under the assumption of a linear system. In the future, we will delve deeper into integrating
a non-linear prediction algorithm and evaluate the resulting challenges of such systems. Notwithstanding this, this paper is the first that consider a prediction-based connectivity solution in HDT by leveraging the potential prediction capabilities of digital twins and thus provide a strong foundation for many future works.

REFERENCES


