Intelligent Digital Twin Communication Framework for Addressing Accuracy and Timeliness Tradeoff in Resource-Constrained Networks

Lal Verda Cakir, *Graduate Student Member, IEEE*, Craig J. Thomson, *Member, IEEE*, Mehmet Özdem, *Member, IEEE*, Berk Canberk, *Senior Member, IEEE*, Van-Linh Nguyen, *Member, IEEE,* and Trung Q. Duong, *Fellow, IEEE*

***Abstract*—The accuracy and timeliness tradeoff prevents Digital Twins (DTs) from realizing their full potential. High accuracy is crucial for decision-making, and timeliness is equally essential for responsiveness. Therefore, this tradeoff in DT communication must be addressed to achieve DT synchronization. Previous studies identified the issue but considered the problem as maximizing data transfer, which is infeasible due to resource constraints. To facilitate this, we quantify accuracy and timeliness as E and ϕ and define the problem as joint minimisation. We then introduce the Intelligent DT Communication (IDTC) Framework to solve the problem, which includes machine learning-based Predictive Synchronization (PS) and DT synchronization management (DTSYNC) protocol. Here, PS uses imputation and forecasting to generate future values, which are utilized to update DT at the projected time points. This mechanism of PS enables lowering E and ϕ of the communication. Subsequently, we utilize the DTSYNC to control synchronization and optimise the twining frequency f_t. We evaluate the proposed framework using a public dataset and compare its performance with several state-of-the-art studies in a real-world scenario. Evaluation results indicate that IDTC outperforms the existing methods by 80% for E and 84% for ϕ while enabling f_t adjustment, resulting in 3.8 times goodput improvement.**

***Index Terms*—intelligent communication, machine learning, digital twin, synchronization, resource management**

I. Introduction

DIGITAL Twins (DTs) are crucial in a variety of industries, e.g., manufacturing plants, healthcare, and autonomous vehicles. DTs allow intelligence and autonomy by offering unique capabilities, such as anticipating what-if situations and making high-precision decisions in virtually real-time [1]–[6]. These applications are built based on the assumption that DT is a replica that is accurate and real-time [7]. In this approach, DT may be seen as a real-world environment without the constraints of restricted access and control. This unlimited access feature enables the development of several cutting-edge applications, particularly when developing test prototypes [8], which is expensive and time-consuming. These require DTs to be accurate and timely simultaneously [9]. However, the synchronization mechanisms and communication protocols to address DTs' accuracy and timeliness tradeoff matter have not yet been well-explored.

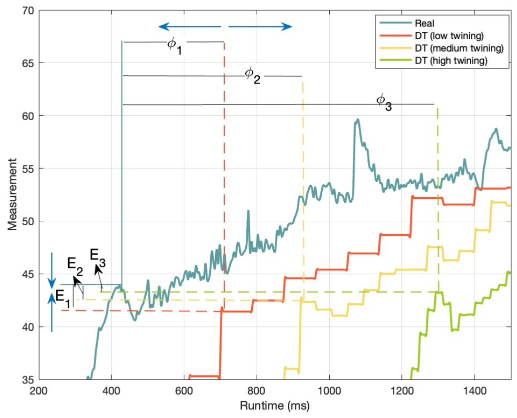


Fig. 1: Accuracy and timeliness tradeoff identified using data from [10]

Generally, DT communication has a tradeoff between accuracy and timeliness caused by resource constraints in the network, preventing DTs from being deployed in practice for now. Figure 1 compares real-world and DT based on the frequency of updates referred to as high/medium/low twining. The DT with a high twining setting may significantly delay the process. In contrast, the DT operates more timely with lower accuracy in the low twining setting. There is a negative correlation between the accuracies ($E_1 > E_2 > E_3$) and the delays which result in DT representation shifts ($\phi_1 < \phi_2 < \phi_3$). Here, accuracy refers to the precision of the DT to replicate the physical entity, which is essential for its operations. The timeliness refers to how fast the DT can provide this

L. V. Cakir is with the School of Computing, Engineering and Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK, and also with the BTS Group, Istanbul, Turkey (e-mail: lal.cakir@napier.ac.uk, verda.cakir@btsgrp.com).

C. J. Thomson is a lecturer in the School of Computing, Engineering and Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK (e-mail: C.Thomson3@napier.ac.uk).

M. Özdem is with Turk Telekom, Turkey (email: mehmet.ozdem@turktelekom.com.tr).

B. Canberk is a professor in the School of Computing, Engineering and Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK (e-mail: B.Canberk@napier.ac.uk).

V. L. Nguyen is with the Department of Computer Science and Information Engineering, National Chung Cheng University (CCU), Taiwan (e-mail: nvlinh@cs.ccu.edu.tw).

T. Q. Duong is with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada, and also with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN Belfast, UK (e-mail: tduong@mun.ca).

replication. At this point, ensuring accuracy and timeliness creates competing objectives due to the bottleneck created by the resources. Resolving this requires effectively managing the network resources by improving the synchronization process and communication protocols.

The DT synchronization has been referred to in the literature as a clock alignment problem, as in [11]. Moreover, authors in [12] propose a control flow mechanism to optimize the clock reconfiguration. The study [13] models the clocks and identifies the time of activating reconfiguration. Meanwhile, the authors in [14] propose a data collection resampling on the clocks' models. While the research may enhance the temporal alignment of the data, focusing just on timestamps undermines the DT synchronization issue. DTs require perfect synchronization. This means a full alignment between the state of the DT and the corresponding real world. The objective is not only to ensure the accuracy of the timestamp and measurement pair but also to establish synchronization at the moment these pairs are generated at both locations, hence ensuring real-time data. Therefore, it is important to consider DT synchronization in a more holistic approach.

Assessing whether this synchronization is attained is not a trivial task. DTs have a disruptive effect on the evaluation metrics used to analyse any synchronization methodologies' performance. Here, the comparison is not just between a value and a baseline that diverges from each other based on a trend, as in the clock problem [15]. In DT synchronization, when the DT representation is compared with the baseline, the respective value for time points and the time itself must be compared. Furthermore, these elements are greatly influenced by an intricate interplay between the efficiency of transmission, processing, and storage, which might exhibit unpredictable fluctuations [16]. Therefore, before considering how to perform this synchronization and construct the communication, we should embark on quantifying its performance.

Multiple studies have attempted to tackle this problem by optimizing the transfer of data at very high frequencies. However, this approach has many flaws [17]. First, with bandwidth constraints, it would be unable to effectively complete this data transfer due to network congestion [18]. This phenomenon was noted in the research when the authors analyzed the performance of DT synchronization. The study also revealed that when the frequency increased, it led to a decline in synchronization quality [19]. To counteract this, selecting a set of devices each time or the data within the devices were proposed in [20], [21]. Moreover, in [22], the data has been disregarded at the IoT gateway if new data has been generated. Another approach explores gathering data from all and then selectively updating the DT in [23]. While these approaches improve the performance metrics themselves, they overlook all devices should be synchronized to achieve synchronization and the DT's dependence on a large amount of data for creating generalized models [24]. From another perspective, [25] proposes minimising delay by scheduling, reducing communication delay. However, this forces the data to wait to be transmitted, which reduces its timeliness.

The performance implications of these challenges are not only reserved for the resource-constrained networks. While improvements in the network's capacity may bring DT closer to synchronization, this approach fails to consider other aspects. Firstly, higher-frequency measurements generate a lot more data, which means more data processing and storage needs, as well as more network capacity [26]. Secondly, in situations when resources are limited, the increased energy consumption caused by increasing capacity is a cause for worry about sustainability [27]. Additionally, the financial burden of extensively upgrading infrastructure may outweigh the benefits. Thus, in order to communicate data while appropriately prioritising the aims of synchronization, the focus should be given to the protocols.

There have been great advancements in improving the latency in the 6G network and Time Sensitive Networks (TSN) areas, which have also been considered as the enablers of DT [28]. However, while these will benefit improving the synchronization performance, they can not address the complexities of managing data flow and control in DTs but can only provide the infrastructure. Because they lack built-in mechanisms to manage DTs, they might need manual programming and customization to fit specific needs. Considering these, synchronization management protocols are necessary.

In short, solving this tradeoff between accuracy and timeliness in DT communication is challenging. To the best of our knowledge, this issue has not yet been solved in the literature. Therefore, this study aims to address the core question *'How to measure and optimize the accuracy and timeliness of synchronization at DT communication in resource-constrained networks?*. For this, we propose a data-centric solution, namely the Intelligent DT Communication (IDTC) Framework. Respectively, we contribute to the literature as follows.

- We identify DT communication's accuracy and timeliness tradeoff in resource-constrained networks. Then, we define DT synchronization as being in an equivalent state through the runtime rather than containing the same timestamp and value pairs. Based on this definition, we measure the performance of ϕ, which stands for timeliness, and E, which stands for accuracy, respectively. Following this, we express synchronization as a joint minimization problem.
- To solve the joint minimisation problem, we eliminate the inherent tradeoff between E and ϕ by incorporating a novel Predictive Synchronization (PS). We also propose a machine learning algorithm to forecast time-series data and calculate future measurements. These anticipated changes are proactively inserted, leading to an average increase of 80% for E and 84% for ϕ. This demonstrates the ability to eliminate the effect of resource constraints on performance.
- For the communication between DT and the real environment, we introduce an application layer protocol named DT synchronization management (DTSYNC), defining the data and control flows in the synchronization process. Using DTSYNC, we incorporate f_t optimization that dynamically adjusts f_t based on E and ϕ. By doing so, we pave the way for DT communication up to 3.8 times more efficiently than the existing methods. Correspondingly,

this ensures that the proposed framework can adjust its operations to cater to the limitations in communication.

The rest of this paper is organised as follows. Section II presents the system model and formulates the problem of synchronization of DTs. Section III presents the proposed IDTC Framework. Section IV provides the performance evaluation. Lastly, Section V concludes the paper by summarizing and discussing future directions.

II. The System Model and Problem Formulation

This section presents our system model for DT and the formulation of the synchronization problem in DT. The challenges of solving the problem are also discussed.

A. System Model

We assume that a DT system consists of two planes, real and virtual [29], as depicted in Figure 2. The real plane comprises devices, namely $D = \{D_1, D_2, ..., D_n\}$, where n represents the number of devices. The devices can be sensors to monitor the surrounding environment. The monitoring is done by sampling a feature of the environment, which we denote $g(t)$. Each device is represented with the tuple $< X, T, f >$, where X is the measured data ($X = g(T)$), T denotes measurement times, and f indicates the measurement frequency. Also, we assume that f_0 serves as a baseline measurement frequency, i.e., the lowest frequency at which measurements have to be collected in order to derive $g(t)$. In other words, if the sample size is less than f_0, we will only have a partial view of the environment. The dynamic nature and properties of the observable environment are then connected to the parameter f_0. Moreover, the DT device, which is in the virtual plane, is denoted as $\hat{D} = \{\hat{D}_1, \hat{D}_2, ..., \hat{D}_n\}$ and represented with the tuple $< \hat{X}, \hat{T}, \hat{f} >$, where $\hat{X}$ represents the data in DT storage, $\hat{T}$ denotes data creation time at DT side, and $\hat{f}$ indicates the frequency of this creation. Here, the $\hat{T}$ represents when a particular data point from D became part of $\hat{D}$ rather than the timestamp of the data. Moreover, the $\hat{X}$ may also be generated by DT through operations such as prediction or forecast. In this context, we define the frequency of the data communicated from D to $\hat{D}$ as f_t where the D is sampling data with f and $\hat{D}$ has its' data with $\hat{f}$. Consequently, the goal of the DT is to generate a signal $h(t)$ that is a copy of the original $g(t)$.

In light of the modeling, we define that $g(t)$ and $h(t)$ are in synchronization if they produce the same result. Here, the $g(t)$ and $h(t)$ are step functions where the function value changes at specific points in time. As the Ds sample the environment, each sample provides a value to the $g(t)$, and this value is communicated to $\hat{D}$, resulting in a value change in $h(t)$. Consequently, these functions hold constant values between sampling/update points. The equivalence of these focuses on the real-time alignment, as mentioned in Section I, and goes beyond just storing the data and timestamp pairings to generate the $h(t)$. Here, the t at both functions represents the actual runtime instead of the timestamp. Therefore, to evaluate this synchronization's performance, we should consider the runtime behaviour of both signals and formulate the performance accordingly. To solve this problem, we propose to calculate the accuracy of DT between times $[t_i, t_j]$ as expressed by

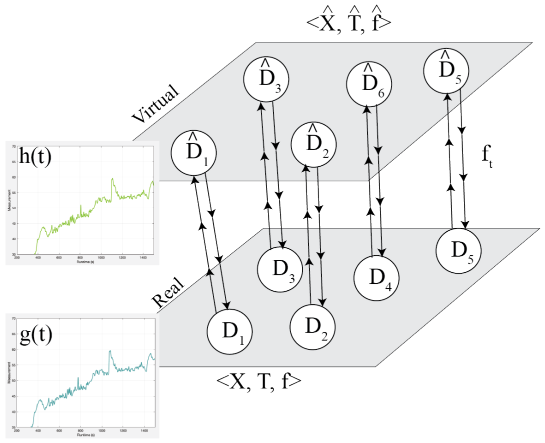


Fig. 2: Our system model for DT and the illustration of the synchronization problem in DT, where DT aims to generate a signal $h(t)$ that is a copy of the original $g(t)$.

$$E = \sqrt{\frac{1}{t_j - t_i} \int_{t_i}^{t_j} g(t)^2 - h(t-\phi)^2 dt}, \quad t_j > t_i \tag{1}$$

where ϕ is the time shift of the trajectory of $g(t)$ when represented on DT. This ϕ represents the timeliness of the DT and is formulated by

$$\phi = \arg\min_{\phi} \int_{t_i}^{t_j} g(t) - h(t-\phi) dt, \tag{2}$$

Given the step function nature of $g(t)$ and $h(t)$, these integrals can be made solvable by decomposing into sums over these intervals. For E:

$$\int_{t_i}^{t_j} g(t)^2 - h(t-\phi)^2 dt = \sum_{k=t_i}^{t_j} (g(t_k)^2 - h(t_k - \phi)^2)(t_{k+1/\hat{f}} - t_k) \tag{3}$$

Similarly, the integral in ϕ is decomposed into:

$$\int_{t_i}^{t_j} g(t) - h(t-\phi) dt = \sum_{k=t_i}^{t_j} (g(t_k) - h(t_k - \phi))(t_{k+1/\hat{f}} - t_k) \tag{4}$$

B. Problem Formulation

We assume that E is representative of the accuracy and ϕ of the timeliness of DT. The synchronization problem can then be expressed by

$$\begin{aligned} \arg\min_{f_t} \quad & E, \phi \\ \text{s.t.} \quad & f_t \le f \le \hat{f} \\ & f = f_0, \\ t_j > t_i & \end{aligned} \tag{5}$$

where $f = f_0$ constraint is applied to ensure that all the essential data capture $g(t)$; without this, the system would not have the actual baseline, and E would suffer a miscalculation. In addition, the time shift, denoted by ϕ, depicts the delays that were experienced throughout the synchronization procedure. Take ϕ as an example; it may be written as $\delta_C + \delta_W + \delta_P$ if synchronization is implemented via data collection solely. In this context, δ_C represents the delay in transmission, whereas δ_P represents the delay in processing at DT. Depending on the occupancy in the network and f_t, there will be an additional delay denoted as δ_W, which corresponds to how much the measurement waits before being transmitted. This joint minimization problem may have an optimum solution, which would be determined based on the resource constraints of the communication and processing capabilities. However, the optimal point would not yield synchronization. Eliminating this tradeoff is necessary to get synchronization. On the other hand, finding the real equivalency of $g(t)$ and $h(t)$ requires addressing the accuracy and timeliness tradeoff.

III. Intelligent Digital Twin Communication (IDTC) Framework

This section presents our proposed Intelligent DT Communication (IDTC) Framework that includes Predictive Synchronization (PS), which leverages machine learning (ML) for synchronization, and DT synchronization management (DTSYNC) protocol, which defines data transfer mechanisms and f_t optimization.

A. Predictive Synchronization (PS)

The predictive synchronization (PS) is proposed to compensate for the delays contributing to the DT's time shift (ϕ), which consequently affects the accuracy E. This is done by proactively inserting the expected measurements into the DT at certain intervals. This technique allows us to shift the synchronization process from a data-collecting issue to a modeling problem, which helps to tackle the tradeoff between accuracy and timeliness. The proposed PS is performed as given in the Algorithm 1 by predicting measurements using the imputation and forecasting models. Here, the data gathered from devices is transformed into time series data with $1/\hat{f}$ intervals using the imputation model. This technique is necessary for setups where f and $\hat{f}$ differ from f_0 and they also may be changed during the runtime. This causes the data to have irregular intervals. However, most forecasting methods require regular interval data. In addition to this, the E and ϕ calculations also operate on equally interval data. Therefore, to acquire complete time-series data, the imputation is needed. Then, the forecasting model uses these imputed time-series data to calculate future measurements of D. These forecasts are compared with the incoming measurement in lines 5-7. If the expected measurements deviate from the actual measurements, the DT storage is corrected, and the model is retrained accordingly in lines 8-11. The forecasts are generated at every $1/\hat{f}$ time for the next step and are scheduled to be written at the time they represent in lines 14-19.

Algorithm 1 Machine learning-based predictive synchronization algorithm

```
1: Initialise $Queue$, $E_{th}$
2: Sync are done with $f_t$ and incoming data placed in $Queue$
3: while true do
4:     if $Queue$ not empty then
5:         $x[t_i : t_j]$ ←Retrieve from $Queue$
6:         $\hat{x}[t_i : t_j]$ ←Retrieve from $Storage$ forecasts for $[t_i t_j]$
7:         $e \leftarrow$ Calculate $E$
8:         if $e > E_{th}$ then
9:             Apply Backpropagation on Models
10:            Update($x$,$[t_i, t_j]$)
11:        end if
12:    end if
13:    $timer = Time()$
14:    if $0 \equiv timer \pmod{1/\hat{f}}$ then
15:        $x$ ←Retrieve from $Storage$ real values with $\hat{f}$
16:        $\hat{x}$ ←Impute($x$)
17:        $\hat{x}_{timer+1/\hat{f}}$ ←Forecast(($timer + 1/\hat{f}$))
18:        Schedule(Write($\hat{x}_{timer+1/\hat{f}}$), $timer + 1/\hat{f}$)
19:    end if
20: end while
```

Here, actual data, forecasts, and the DT representation are managed separately within the DT storage. The actual data and forecasts are stored in a historical database, while the DT representation is kept in a time-series database. The retention period for the time-series database is $t_j - t_i$, which corresponds to the period during which the evaluations are made. When a forecast is calculated, it is stored in the forecast and the DT representation storage. Then, when new data is retrieved, the write is done simultaneously with the actual data and DT representation storage. Here, the writing to the DT representation constitutes the correction. By this approach, we ensure that both actual and forecasted data are held historically, and the DT representation is used for the D.

When applying PS, a variety of models for imputation and forecasting present in the literature can be utilized. For this, we evaluate a set of candidate models in Section IV-C2 and decide upon the multi-directional recurrent neural network and Long Short Term Memory based on their performances. Other algorithms can also replace these models based on their specific performance in the use case.

On the other hand, using forecasts for DT synchronization to compensate for the delays can also affect performance. The dynamics of the real environment can change, and the prediction method should be able to adapt to this. To overcome this, the PS mechanism uses the data received from the D to retrain its models repeatedly.

B. Digital Twin Synchronization Management (DTSYNC) Protocol

Besides using the predictions for synchronization, we propose a novel protocol, namely Digital Twin synchronization management (DTSYNC). DTSYNC defines mechanisms to form data, control flows, and manage synchronization through

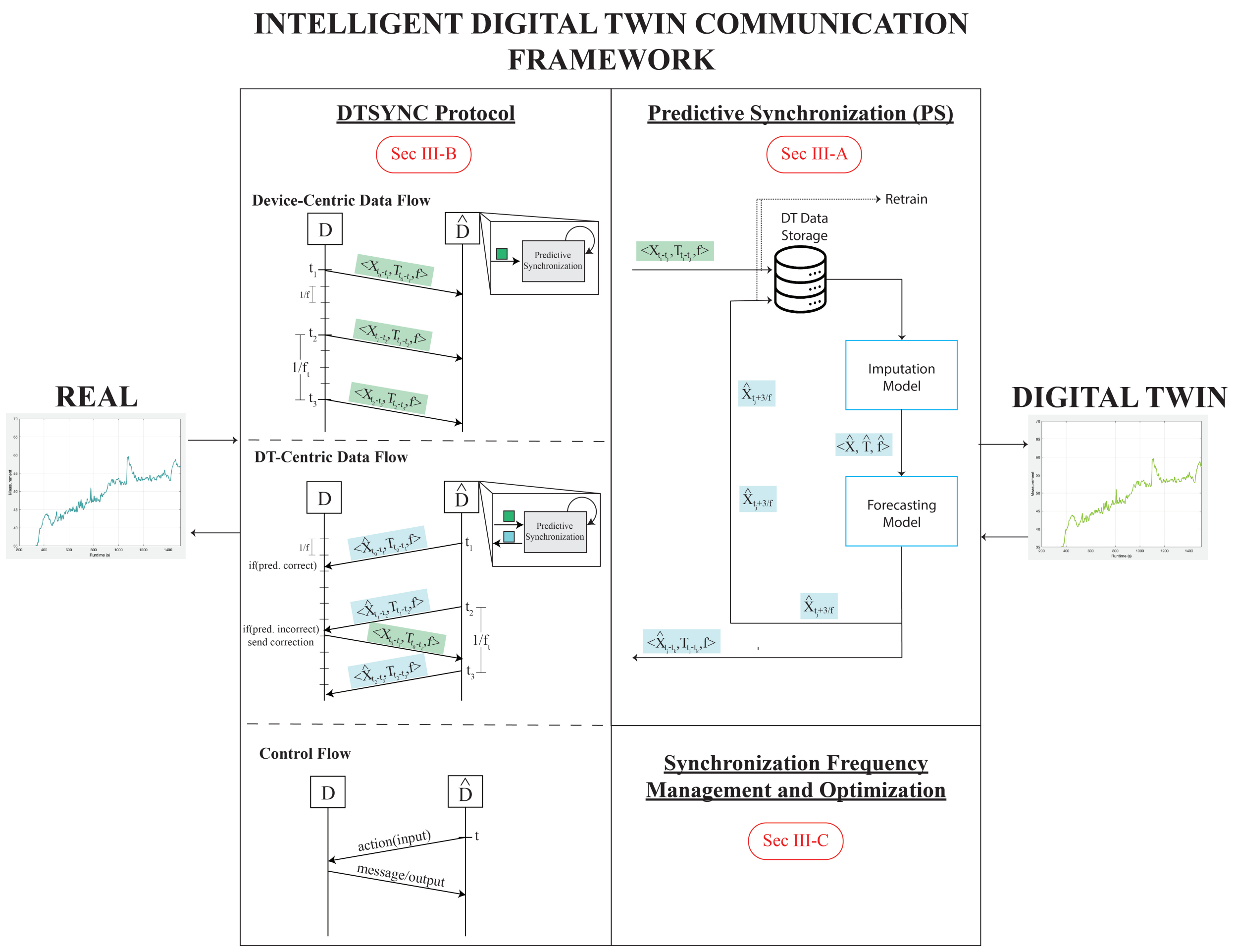


Fig. 3: Our proposed Intelligent DT Communication (IDTC) framework. IDTC includes Predictive Synchronization (PS), which leverages machine learning (ML) for synchronization, and Digital Twin synchronization management (DTSYNC) Protocol, which defines data transfer mechanisms and f_t optimization.

adjusting f_t. In this protocol, we create an approach that leverages the predictions to reduce the number of data exchanges and optimizes this based on the accuracy of the models and network conditions. Moreover, the DTSYNC protocol works with the PS to solve the accuracy and timeliness tradeoff. PS minimizes E and ϕ under a set f_t setting, selecting the ideal f_t based on the environment dynamism and PS performance.

The proposed protocol can be layered on application or transport layer protocols. This allows half-/full-duplex communication to accommodate data and control flows and persistent connection to ensure continuous communication. Besides, it also helps to avoid the overhead of establishing new connections. Furthermore, the accuracy of DT may be compromised by any loss of data or faults in transmission, while the distribution of unsequenced data might lead to misunderstanding of data, hence increasing the complexity of DT. Considering these requirements, the Transmission Control Protocol (TCP) is chosen for its error-handling retransmission and sequenced data delivery mechanisms. Based on these, we identify the candidate application protocols as HTTP, Websocket [30], NETCONF [31], and RESTCONF [32] when used under the keep-alive settings.

1) Data Flow: The purpose of establishing data flows is to ensure the synchronization of measurements obtained at the device with the DT side. Here, the devices operate based on f and transfer data based on f_t as discussed in Section II. In DTSYNC, this process can be performed in a device-centric or DT-centric manner, as depicted in Figure 3.

The device-centric data flow involves the transmission of measurements from devices (D) to DT ($\hat{D}$) during the last time frame ($1/f_t$). $\hat{D}$ utilizes the incoming data following the methodology outlined in Section III-A. The error of PS for each device is evaluated on the DT side; thus, all related processing is performed on one side. The distribution of DT-centric data flow occurs via the transmission of forecasts from the previous window to D for their approval. If the values correspond to the data at D, no more actions are executed; otherwise, the adjustment is sent. The use of a distributed approach may provide scalability in cases where the precision of PS operations is high. On the other hand, it can require at most twice the data transfer than device-centric when the PS performance is low due to additional corrections. Therefore, these two types must be considered based on the system requirements, constraints, and long-term objectives.

2) Control Flow: The control flow has a remote procedure call paradigm initiated by DT, sending the action with relevant input parameters. Upon receipt, the device initiates a response and generates the corresponding message or output. In this context, the action might include reconfiguring the device or initiating an activity, such as restarting.

C. Synchronization Frequency Management and Optimization

The data flow is used to synchronize at regular intervals of $1/f_t$ by initiating the transfer of measured/predicted data. Moreover, the transferred data is formed with the frequency of f and includes the data within the $1/f_t$ window as depicted in Figure 3. Furthermore, the control flow is used to establish the necessary configurations. In this, f is a parameter whose' selection is crucial in ensuring the correctness of DT, as explained in Section II. It should be set to f_0, which was initially determined based on the environmental characteristics. f_t affects the timeliness of DT because a measurement can wait up to $1/f_t$ seconds to be included in the data transfer, correlating to δ_W. Also, this factor affects the message size and the number of transfers done over time.

The amount of the meaningful data used to share would not vary depending on f_t. However, raising f_t will result in more messages being sent, which will raise the protocol overhead. This is because the number of bits accounting for one timestamp and the measurement pair is considerably smaller than protocol header information. As a result, in the case of high network usage, the increase in f_t will decrease the goodput, increasing δ_C. In other words, the number of measurements that reach $\hat{D}$ in one period will decrease when f_t increases. On the other hand, When a lower f_t is utilized, the flow's quality of service (QoS) improves, but the δ_W factor increases since the measurement has to wait at most $1/f_t$. While the PS's goal is to compensate for this, determining how much compensation can be achieved is difficult due to the dynamic nature of the real environment, changes in behaviour over time, and model performance. Additionally, using the predicted values in future predictions within the PS might lead to an accumulation of errors, making the results deviate.

In summary, optimization should also consider the performance of the PS mechanism and take network conditions into account. For instance, higher values on f_t can be more beneficial in high-bandwidth networks by optimizing the goodput and latency, whereas low-bandwidth or congested networks might require lower values to minimize the overhead caused by the retransmissions due to disruption in the communication and maintain performance. Given these considerations, the f_t must be selected depending on model accuracy, resource usage, and communication QoS. However, this requires comprehensive information about each factor, which is often impossible or impractical. For instance, the QoS of the flow may be affected by network occupancy, topology, and number of hops. The f_t selection problem is then similar to the challenge of adjusting the TCP window size. In all cases, judgements must be taken based on limited knowledge in order to balance opposing goals. For example, increasing f_t can enhance accuracy and real-time performance but may exacerbate congestion issues due to increased message exchanges. Similarly, enlarging the TCP window size can improve network throughput but may lead to congestion and decreased overall performance.

To address the challenge, we perform the f_t adjustment based on a similar additive increase/multiplicative decrease approach, which is expressed by

$$f_t = \alpha f, \tag{6}$$

where $\alpha_0 = 1$ and

$$\alpha_{n+1} = \begin{cases} max(10^{-4}, \frac{\alpha_n}{\alpha_n+1}) & E < E_{th} \land \phi < \phi_{th} \\ max(10^{-4}, \frac{\alpha_n}{\alpha_n+1}) & E < E_{th} \land \alpha_n = 1 \land \phi > \phi_{th} \\ min(1, \frac{\alpha_n}{1-\alpha_n}) & E < E_{th} \land \alpha_n \neq 1 \land \phi > \phi_{th} \\ max(1, \frac{2}{\alpha_n}) & E > E_{th} \end{cases} \tag{7}$$

Here, each case corresponds to the following

- Case 1: The DT operates with an error lower than the threshold ($E < E_{th}$), meaning the operation is done accurately. And the time shift of DT is at minimum ($\phi < \phi_{th}$). This might happen in the following two cases:
  - PS is not utilized, meaning $\alpha_n = 1$. In this case, the network can accommodate the data flow. Lower α to preserve network resources and employ PS instead.
  - PS is utilized, meaning $\alpha_n < 1$. Here, predictive mechanisms compensate δ_C and δ_W, resulting in low ϕ. Lower α to preserve network resources and use PS more.
- Case 2: The DT can operate accurately, the PS is not employed, and the time shift of DT is higher than the threshold. In this case, the network cannot accommodate this data flow. Lower α to lower ϕ and employ PS instead.
- Case 3: The DT can operate with maximum accuracy using PS ($\alpha_n < 1$), but delays cannot be compensated ($\phi > \phi_{th}$). The window size may be causing a bottleneck on the DT side, meaning that the ϕ is affected by δ_P. Increase α to preserve DT resources and increase timeliness.
- Case 4: The DT cannot operate accurately. Regardless of ϕ, more training is needed for the PS model. Increase α to use actual values instead of predictions.

IV. Performance Evaluation

This section describes datasets, evaluation settings, measurement metrics, and a detailed analysis of the system performance of the proposed framework.

A. Dataset

To measure the performance of the proposed framework in a real scenario, we use the Sofia Air Quality dataset [10]. The dataset chosen is one of the few datasets in which data is simultaneously collected from different devices. This is essential for simulating a real-world DT scenario and performance evaluation of IDTC framework. The dataset contains data collected from outdoor sensor data in Sofia from July 2017 to July 2019. The dataset includes temperature, humidity, and pressure measurements. New sensors have been added to the dataset monthly throughout the data collection period, totalling up to 459 sensors. To maintain consistency, we chose sensors that are present across the dataset, resulting in 15 different sensors. The average sample period for these sensors is 2.5 minutes, with a standard deviation of 30 seconds. As a result, the environment's f_0 is set at $f_0 = 2.5$ per minute. D uses this data to mimic data production. Here, the 2.5-minute interval is large; the performance evaluation focuses on millisecond-level differences in timeliness and investigates the improvements.

B. *Simulation environment and measurement metrics*

To provide a comprehensive analysis of the performance of the proposed IDTC framework, we perform testing for each component: the PS, DTSYNC Protocol, and f_t optimization mechanism. The simulation environments were constructed using Python 3.10 [33] and the OMNET++ network simulator [34]. PS models are built using the Sklearn, Keras, TensorFlow, and PyTorch libraries. The corresponding experimentations are performed by inputting the model with the data with time intervals based on the experiment setting. When performing analysis on PS solely, the network is not simulated. As for the DTSYNC protocol and f_t optimization mechanism, data communication is simulated using OMNET++.

The network simulation is set up with two access points and 15 stations that are mobile within their coverage area. In the topology, the access points are linked to a router that connects to the cloud network, with configurable latency, transmission speed, and error rate. At the other end, the node where the DT is installed is linked to a router, which is then linked to the cloud network. Furthermore, the DTSYNC protocol is implemented with HTTP, Websocket, NETCONF, and RESTCONF. Further information on the experimental settings may be found in the subsections below.

C. *Evaluation results of the predictive synchronization scheme*

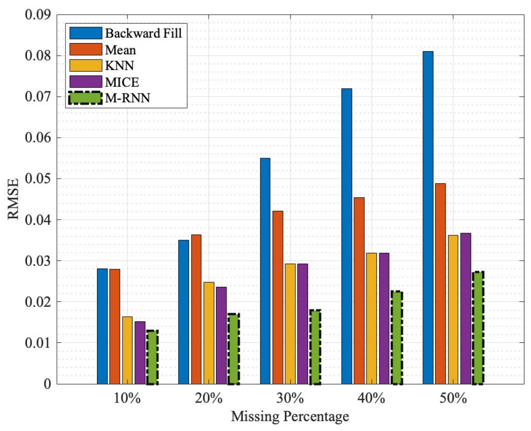


Fig. 4: RMSE Comparison of Imputation Models

1) Imputation Model: We evaluate the performance of candidate imputation methods of PS. Here, we select the candidate imputation methods as backward filling and mean value imputation [35], multiple imputation by chained equations (MICE) [36], k-nearest neighbours (KNN) [37] and multi-directional recurrent neural network (M-RNN) [38]. To evaluate, we randomly generate missing values in the one sensor data with the percentages of [10, 20, 30, 40, 50]. Then, we compare the Root Mean Square Error (RMSE) calculated over the scaled values for the results as shown in Figure 4. Here, the missing percentages are created manually to evaluate the models' capability of learning the behaviour of the data through time.

The results show that M-RNN can achieve the lowest RMSE while mean value imputation performs the worst. The M-RNN's performance can be credited to the bidirectional connections enabling the model to learn and leverage the correlations in the time axis and different variables within the data. Furthermore, the KNN and MICE techniques provide remarkably comparable results. Because both approaches rely on logical reasoning between characteristics, they represent the relationship between distinct features. However, since they solely focus on the relation between features, they neglect the correlation in the time and cannot reach the M-RNN's performance.

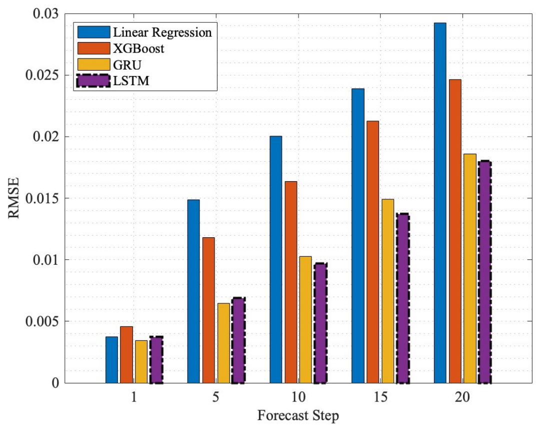
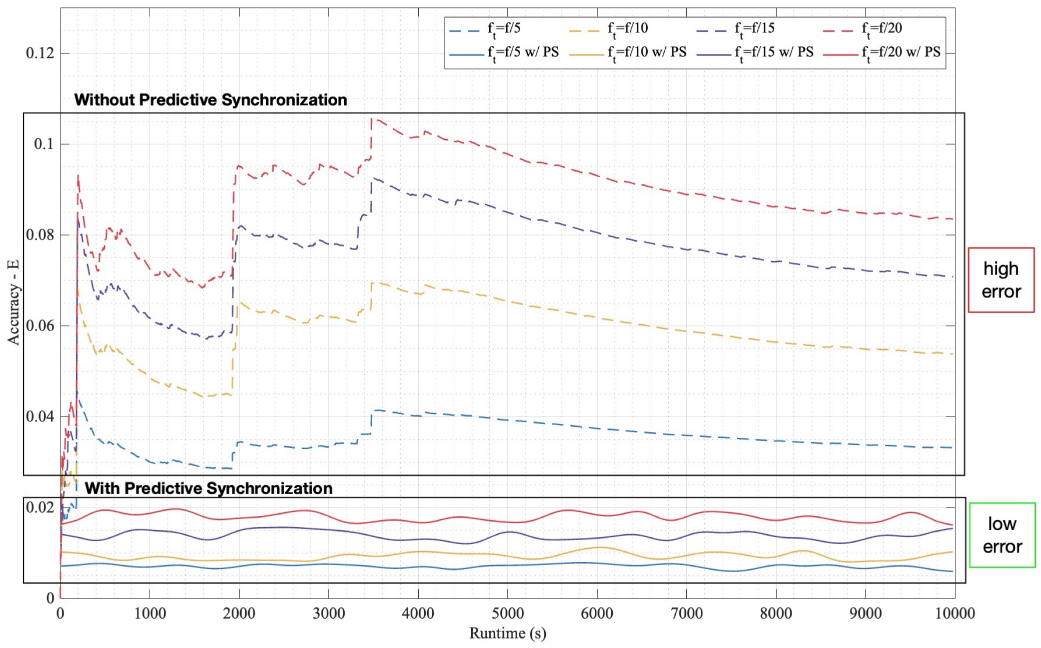
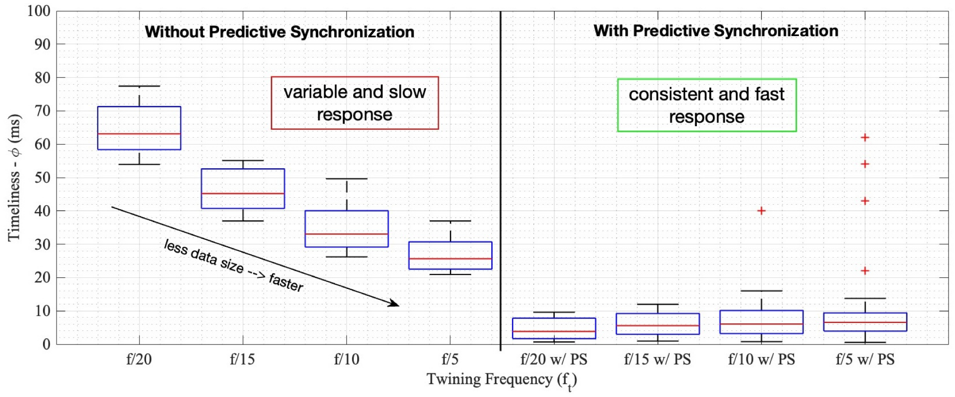


Fig. 5: RMSE Comparison of Forecasting Models

2) Forecasting Model: For the forecasting model, the linear regression, XGBoost [39] and Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models [40] are compared for their RMSE in forecasting future steps recursively. Here, we see a positive correlation between the forecast steps and RMSE. This is because errors accumulate when the model outputs are utilized again as input. Furthermore, the findings suggest that LSTM and GRU perform better than other models, which may be ascribed to the RNN's capacity to capture long-term relationships. We see that LSTM performs marginally better in higher forecast stages. Furthermore, it is worth noting that linear regression and XGBoost were used in this experiment with separate model instances for each feature and sensor. As a result, employing these models for DTs and big scenarios is not scalable or practicable because of the high memory and computing resource needs, maintenance overhead, code complexity, and scalability. Considering these,



(a) Accuracy (E) Comparison



(b) Timeliness (ϕ) Comparison

Fig. 6: Predictive Synchronization (PS) Performance Analysis

we opt to use LSTM in the PS.

3) Predictive Synchronization (PS) Mechanism: In the PS experimentations, the arrival process is simulated according to the sampling intervals in the dataset. Correspondingly, the evaluation is done for the E and ϕ with and without PS under f_t settings of $[f/5, f/10, f/15, f/20]$ and shown in Figure 6. The E is calculated as formulated in Section II-B with the time window 600s. Here, the latest data is taken as the definitive value of DT until the new update is done. Even if the data contain historical values, this assessment excludes retroactive updates. This consideration is based on the DT's definition, which is synchronized during the execution. As a result, the assessment should be focused on its current worth rather than future revisions. Furthermore, these experiments do not contain network simulation, allowing us to concentrate our evaluation simply on the performance of the synchronization process itself, regardless of the variety of possible delays between simulation runs. This impact will be investigated further in the next sections.

In the overall case, the results for E is shown in the Figure 6a. Generally, there is a reverse association between f_t and E. For scenarios without PS, this is because the time between data updates grows as f_t decreases, increasing the amount of change that goes unnoticed. Furthermore, we can observe that the error builds over time and eventually converges. The converges values are highly similar to those obtained using the backward fill method in Section IV-C2. The persistence of the DT's present value until the arrival of fresh data may be ascribed to a phenomenon known as backward filling. On the other hand, PS performs with 80% lower E while having a similar reverse correlation between f_t and E. Here, this reverse correlation is caused by the error accumulation discussed in Section IV-C2. Moreover, E values fluctuate around the forecasting model errors observed in Figure 5, and these fluctuations may be credited to the environment's dynamicity. This dynamicity could have caused the E to deviate, but it is avoided thanks to the retraining in the Algorithm 1.

For sensitive analysis, we evaluate the ϕ in Figure 6b by showing its distribution for each setting. Here, it is observed that the ϕ without PS results in higher and more variable results than PS. These results demonstrate a reverse correlation between f_t and ϕ. This is because as the f_t decreases, the size of the incoming data increases, which requires more time to be processed. On the other hand, in PS mechanisms, the only delay encountered is the write latency to the DT storage. In PS, the necessary calculations and processing are performed beforehand, and the writing process is scheduled. This enables the PS mechanism to compensate for the delays, lowering ϕ by 84% on average. It is important to note that in PS ϕ results, there are some outlier cases in which an extreme ϕ is observed. This may be caused by different processes simultaneously operating on the DT. While the PS case only involves writing incoming data to storage, the PS operations include further access to the storage, such as retrieving historical values. These may increase data backlogs and contribute to ϕ.

D. Evaluation results of DTSYNC with various protocols

The DTSYNC is evaluated by implementing it using HTTP, NETCONF, RESTCONF, and WebSocket. We implement HTTP, NETCONF and RESTCONF in persistent configuration, and Websocket has this inherently. All of these application layer protocols follow a server-client architecture. Therefore, in the experimentation of device-centric data flows, the DT operate as a server because the data flow will be initiated from the device. As for the DT-centric data flow, setup is done with DT as the client because the DT will initiate the synchronization process. Then, accordingly, we implement the DTSYNC protocol defined in Section III-B

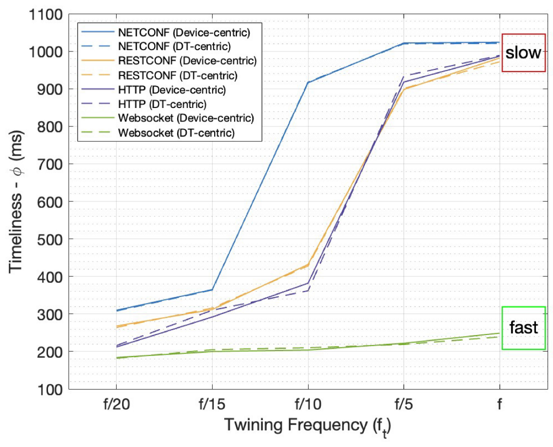
In Figure 7a, we compare the achieved timeliness ϕ results with correction probability $p = 0$. Here, the correction probability represents the probability that D will decide that the predictions are incorrect compared to the actual data it has measured. Moreover, this ϕ correlates to the communication delay (δ_C) as discussed in the Section II-B. The figure shows that the data flow configurations perform highly similarly since the amount of data transferred is the same when corrections are not needed. Here, we observe that the correction probabilities do not affect the ϕ results even though the corrections would also have a processing cost. This is due to the fact that the timeliness for a timeframe is calculated for the time when this correction has not occurred yet. In other words, this retrospective update is part of the future timeliness calculation that will occur when the correction operation has already ended. Overall, results show WebSocket outperforms because the data is directly sent through one TCP channel without following a request and response pattern. On the other hand, other protocols in this persistent configuration also use one TCP channel but have a higher protocol overhead. This increase in the number of bits sent leads to the rise of ϕ. Considering this, we implement DTSYNC using WebSocket protocol.

With diversity consideration, we evaluate the ϕ of device-centric and DT-centric with correction probabilities of $p = [0, 0.25, 0.5, 0.75, 1]$ as shown in Figure 7b. Here, we observe that as these percentages increase, ϕ in the DT-centric worsens while the device-centric operation is consistent throughout. Moreover, we also consider related resource consumption by analysing the average CPU utilisation at the DT in Figure 7c. Here, the device-centric operation has also performed highly similar under all p scenarios. This is because most of the processing is performed for the related data exchange and the initial processing, and these are not changed by p in device-centric. As a result, this outweighs the memory update operation for correction in device-centric. As seen, under a low correction probability (p), the DT-centric requires lower resources than the device-centric. This difference lowers as the correction probability increases, revealing that the device-centric can be preferred during the training period. Then, the operation can be continued in a DT-centric manner to preserve resources. This flow type change can be implemented without closing the current session in WebSocket, thanks to its bidirectional nature.

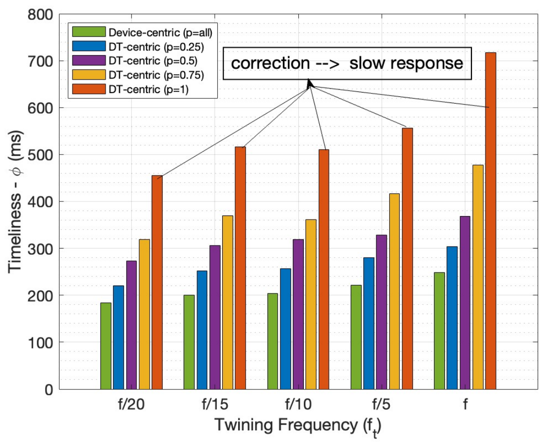
PS and DTSYNC experimentation outcomes till this point indicate that using PS with higher f_t's should be preferable since it will yield smaller ϕ and E. However, this comparably slight increase with higher f_t values will require more resources. This is observed in the positive correlation between f_t increase and CPU utilisation in Figure 7c. To further analyse this, we evaluate the efficiency of the network resource usage by comparing the throughput, and the goodput results under different f_t in Figure 7d. Here, as f_t decreases, while throughput declines, the goodput increases and converges to a similar value. This convergence point is where the data of the DTSYNC protocol reaches the threshold of the Maximum Segment Size (MSS) in TCP, thus requiring segmentation. This segmentation reintroduces the overhead the lower f_t was trying to achieve. These results show that the goodput of synchronization can be increased up to 3.8 times. This significant increase is highly important in resource-constrained networks where the bandwidth is limited. By reducing the overhead, a better utilization of resources can be done. This also can lead to improvements in the latency of the communication by minimizing retransmissions and processing delays, thus reducing overall latency in DT synchronization. This efficiency improvement further highlights the need for f_t dynamic optimization.

E. Evaluation results of synchronization frequency optimization scheme

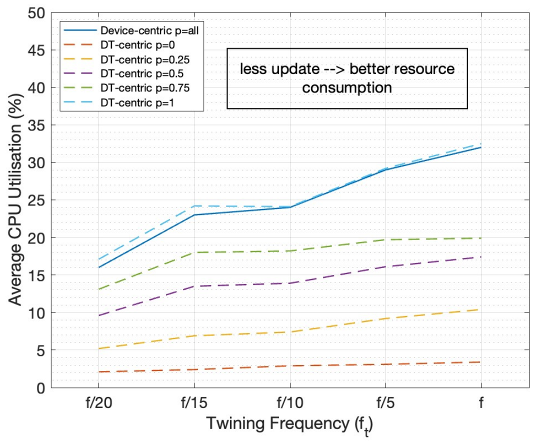
For initial testing, we start the experiment with a non-trained model in PS and implement DTSYNC with device-centric data flow. This model is trained during the runtime, and the synchronization frequency (f_t) is adjusted in response model performance as explained in Section III-C. The threshold values E_{th} and ϕ_{th} of f_t optimization are selected as 0.02 and 15 based on the findings in the Section IV-C3. Although we showcased that PS achieves these thresholds, we start with a non-trained model simulating real-world scenarios where pre-existing data is absent or adaptation to a new environment is



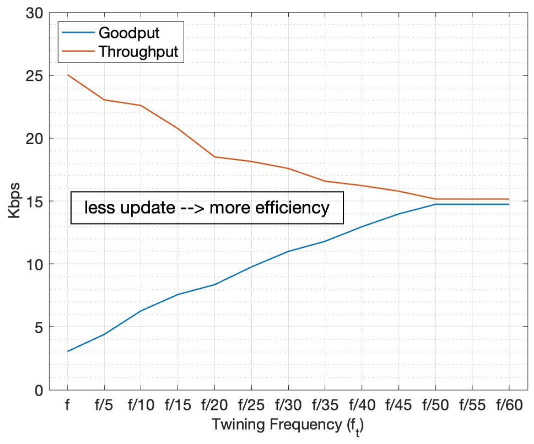
(a) ϕ Comparison of DTSYNC under $p = 0$



(b) Flow Type ϕ Comparison of DTSYNC under $p = 0.25, 0.5, 0.75, 1$



(c) CPU Utilisation Comparison of DTSYNC under $p = 0.25, 0.5, 0.75, 1$



(d) Goodput and Throughput Comparison of DT-SYNC under $p = 0$

Fig. 7: DTSYNC Performance Analysis

needed. By this, we aim to evaluate the performance of the IDTC framework based on the evolution of E and ϕ under f_t optimization. The results are shown in Figure 8. The decision-making in the Equation 7 is applied every 600 steps of the runtime. As seen in the initial phases of the experiment, the algorithm goes back and forth between cases 2 and 4 to test whether the PS mechanism is fit to be applied.

The fluctuation can be avoided by not permitting case 2 to be initiated before the model training loss achieves a certain threshold. However, we do not employ this here to showcase the behaviour when changing environment dynamics. Moreover, the results reach a certain equilibrium after the 3600, which leads to lowering f_t more. Thanks to PS, this does not significantly affect the E and ϕ, so this optimization mechanism can lower f_t. As a result, the communication overhead can be reduced, and efficiency can improve, as discussed in Section IV-D.

F. Opportunities and Challenges of IDTC

Integrating Intelligent Digital Twin Communication (IDTC) frameworks presents numerous opportunities and challenges across various DT applications within different industries. The creation of both accurate and timely representations of the real environment can lead to improvements in operational efficiency and responsiveness. While only time-series IoT data has been used within the scope of this article, the PS mechanism can modified to be applied to different data from fields such as aerospace, automotive and energy. In this, simulations/emulations can be used to predict future conditions, and corrections can be made to its parameters dynamically. Additionally, the framework can be especially leveraged in DTs for network management and be used to intelligently control f_t using proposed performance metrics and data collected on the networks' performance. Correspondingly, there can be future challenges as follows:

- Complexity in Implementation: Deploying an IDTC framework can involve significant complexity in decid-

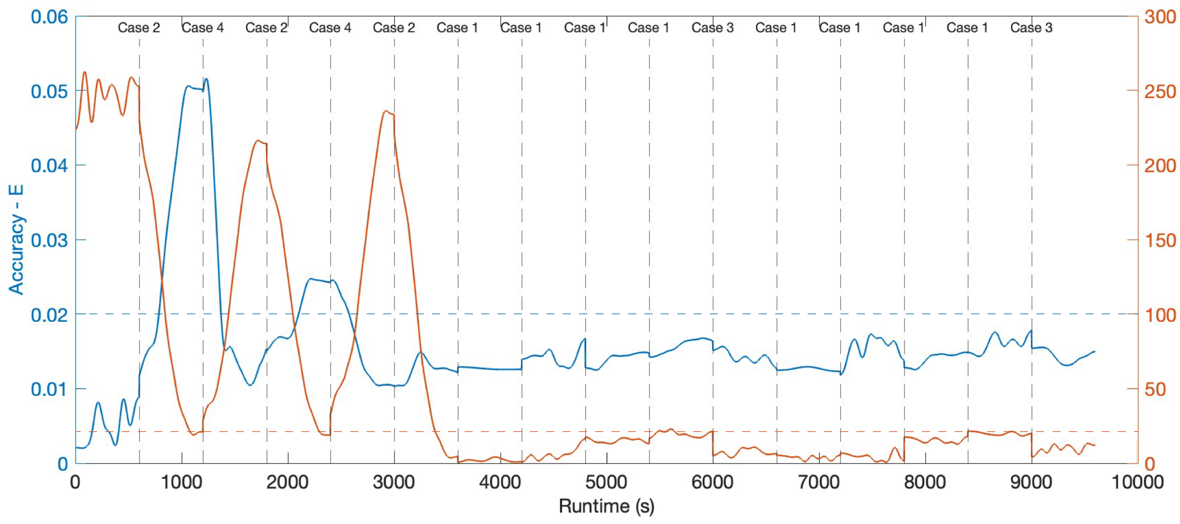


Fig. 8: f_t Optimization

ing on and setting up models. This requires substantial expertise in both the underlying methods and the specific domain.

- Resource Constraints: The benefits of IDTC can come at the cost of increased computational and memory resources on the DT side. While the f_t optimization in the IDTC aims to improve the efficiency of this resource usage, management of the infrastructure can pose challenges.

V. Conclusion

This research examines the tradeoffs between accuracy and timeliness in the context of Digital Twins (DTs). In order to address this difficulty, we formulate the synchronization problem by using the proposed E and ϕ metrics and then propose a novel Intelligent DT Communication (IDTC) Framework with Predictive Synchronization (PS) and DT synchronization management (DTSYNC) protocol. This framework updates the DT data predictively using machine learning to compensate for the delays in resource-constrained networks. By doing this, we eliminate the accuracy and timeliness tradeoff. Furthermore, we establish the DTSYNC protocol, which facilitates the flow of data and regulates the twining frequency (f_t). In this context, the variable f_t is subject to dynamic management in order to minimize the variables E and ϕ, resulting in enhanced efficiency in the use of resources, which enables the 3.8 times goodput improvement. In the future, we plan to explore User Datagram Protocol (UDP)-based protocols in DTSYNC, which can lower delay and resource consumption. Moreover, we aim to analyse the different computing and storage infrastructures' effects on performance and validate the proposed framework in real-world conditions by using large-scale testbeds.

Acknowledgments

This work is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) 1515 Frontier R&D Laboratories Support Program for BTS Advanced AI Hub: BTS Autonomous Networks and Data Innovation Lab. Project 5239903. The work of T. Q. Duong was supported in part by the Canada Excellence Research Chair (CERC) Program CERC-2022-00109 and in part by the U.K. Royal Academy of Engineering (RAEng) under the RAEng Research Chair and Senior Research Fellowship scheme Grant RCSRF2021\11\41. Moreover, L. V. Cakir would like to thank the Deep Mind Scholarship Programme for their support.

References

[1] Z. Tang, D. Chen, T. Sun, L. Zhang, M. Qi, and X. Wang, "Intelligent awareness of delay-sensitive internet traffic in digital twin network," *IEEE Journal of Radio Frequency Identification*, vol. 6, pp. 891–895, 2022.

[2] Z. Wei, S. Wang, D. Li, F. Gui, and S. Hong, "Data-driven routing: A typical application of digital twin network," in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021, pp. 1–4.

[3] J. Deng, Q. Zheng, G. Liu, J. Bai, K. Tian, C. Sun, Y. Yan, and Y. Liu, "A digital twin approach for self-optimization of mobile networks," in *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2021, pp. 1–6.

[4] H. Wang, Y. Wu, G. Min, and W. Miao, "A graph neural network-based digital twin for network slicing management," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1367–1376, 2022.

[5] H. Boche, R. F. Schaefer, H. V. Poor, and F. H. P. Fitzek, "On the need of neuromorphic twins to detect denial-of-service attacks on communication networks," *IEEE/ACM Transactions on Networking*, pp. 1–13, 2024.

[6] D. Van Huynh, V.-D. Nguyen, S. R. Khosravirad, G. K. Karagiannidis, and T. Q. Duong, "Distributed communication and computation resource management for digital twin-aided edge computing with short-packet communications," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3008–3021, 2023.

[7] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Network digital twin: Concepts and reference architecture." [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/

[8] A. Madni, C. Madni, and S. Lucero, "Leveraging digital twin technology in model-based systems engineering," *Systems*, vol. 7, no. 1, p. 7, 2019.

[9] O. Hashash, C. Chaccour, and W. Saad, "Edge continual learning for dynamic digital twins over wireless networks," in *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022, pp. 1–5.

[10] H. Mavrodiev, "Sofia air quality dataset," September 2019, accessed: 2024-01-21. [Online]. Available: https://www.kaggle.com/datasets/hmavrodiev/sofia-air-quality-dataset/

[11] H. Yang, Y. Li, K. Yao, T. Sun, and C. Zhou, "A systematic network traffic emulation framework for digital twin network," in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, 2021, pp. 94–97.

[12] E. Ak, K. Duran, O. A. Dobre, T. Q. Duong, and B. Canberk, "T6conf: Digital twin networking framework for ipv6-enabled net-zero smart cities," *IEEE Communications Magazine*, vol. 61, no. 3, pp. 36–42, 2023.

[13] P. Jia, X. Wang, and X. Shen, "Digital-twin-enabled intelligent distributed clock synchronization in industrial iot systems," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4548–4559, 2021.

[14] ——, "Accurate and efficient digital twin construction using concurrent end-to-end synchronization and multi-attribute data resampling," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 4857–4870, 2023.

[15] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, 2004.

[16] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, G. Karakostas, H. Wu, and X. Shen, "Digital twins from a networking perspective," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 525–23 544, 2022.

[17] S. Mihai, M. Yaqoob, D. V. Hung, W. Davis, P. Towakel, M. Raza, M. Karamanoglu, B. Barn, D. Shetve, R. V. Prasad, H. Venkataraman, R. Trestian, and H. X. Nguyen, "Digital twins: A survey on enabling technologies, challenges, trends and future prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022.

[18] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[19] L. V. Cakir, S. Al-Shareeda, S. F. Oktug, M. Özdem, M. Broadbent, and B. Canberk, "How to synchronize digital twins? a communication performance analysis," in *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2023, pp. 123–127.

[20] J. Zheng, T. H. Luan, Y. Zhang, R. Li, Y. Hui, L. Gao, and M. Dong, "Data synchronization in vehicular digital twin network: A game theoretic approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 7635–7647, 2023.

[21] Z. Yang, M. Chen, Y. Liu, and Z. Zhang, "Optimizing synchronization delay for digital twin over wireless networks," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 9106–9110.

[22] L. V. Cakir, T. Bilen, M. Özdem, and B. Canberk, "Digital twin middleware for smart farm iot networks," in *2023 International Balkan Conference on Communications and Networking (BalkanCom)*, 2023, pp. 1–5.

[23] L. V. Cakir, K. Duran, C. Thomson, M. Broadbent, and B. Canberk, "Ai in energy digital twining: A reinforcement learning-based adaptive digital twin model for green cities," 2024. [Online]. Available: http://arxiv.org/abs/2401.16449

[24] L. Hui, M. Wang, L. Zhang, L. Lu, and Y. Cui, "Digital twin for networking: A data-driven performance modeling perspective," *IEEE Network*, vol. 37, no. 3, pp. 202–209, 2023.

[25] Z. Yang, M. Chen, Y. Liu, and Z. Zhang, "Optimizing synchronization delay for digital twin over wireless networks," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 9106–9110.

[26] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, "Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2230–2254, 2022.

[27] K. Duran and B. Canberk, "Digital twin enriched green topology discovery for next generation core networks," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 4, pp. 1946–1956, 2023.

[28] G. P. Sharma, D. Patel, J. Sachs, M. De Andrade, J. Farkas, J. Harmatos, B. Varga, H.-P. Bernhard, R. Muzaffar, M. Ahmed, F. Dürr, D. Bruckner, E. M. De Oca, D. Houatra, H. Zhang, and J. Gross, "Toward deterministic communications in 6g networks: State of the art, open challenges and the way forward," *IEEE Access*, vol. 11, pp. 106 898–106 923, 2023.

[29] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Digital twin network: Concepts and reference architecture." [Online]. Available: https://www.ietf.org/archive/id/draft-irtf-nmrg-network-digital-twin-arch-04.html

[30] A. Melnikov and I. Fette, "The WebSocket Protocol," RFC 6455, Dec. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6455

[31] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, "Network Configuration Protocol (NETCONF)," RFC 6241, Jun. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6241

[32] A. Bierman, M. Björklund, and K. Watsen, "RESTCONF Protocol," RFC 8040, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8040

[33] "Python 3.10." [Online]. Available: https://www.python.org/downloads/release/python-31013/

[34] "Omnet++ discrete event simulator." [Online]. Available: https://omnetpp.org/

[35] I. Pratama, A. E. Permanasari, I. Ardiyanto, and R. Indrayani, "A review of missing values handling methods on time-series data," in *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2016, pp. 1–6.

[36] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, "Multiple imputation by chained equations: what is it and how does it work?" *Int J Methods Psychiatr Res.*, vol. 20, no. 1, pp. 40–49, 2011.

[37] S. Tak, S. Woo, and H. Yeo, "Data-driven imputation method for traffic data in sectional units of road links," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1762–1771, 2016.

[38] J. Yoon, W. R. Zame, and M. van der Schaar, "Estimating missing data in temporal data streams using multi-directional recurrent neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 5, pp. 1477–1490, 2019.

[39] R. A. Abbasi, N. Javaid, M. N. J. Ghuman, Z. A. Khan, S. Ur Rehman, and Amanullah, "Short term load forecasting using xgboost," in *Web, Artificial Intelligence and Network Applications*, L. Barolli, M. Takizawa, F. Xhafa, and T. Enokido, Eds. Cham: Springer International Publishing, 2019, pp. 1120–1131.

[40] R. A. Rajagukguk, R. A. A. Ramadhan, and H.-J. Lee, "A review on deep learning models for forecasting time series data of solar irradiance and photovoltaic power," *Energies*, vol. 13, no. 24, 2020.

Lal Verda Cakir (Graduate Student Member, IEEE) is pursuing her PhD in the School of Computing, Engineering and the Built Environment at Edinburgh Napier University.

Craig J. Thomson (Member, IEEE) is a lecturer at Edinburgh Napier University, United Kingdom.

Mehmet Özdem (Member, IEEE) currently works as the Director of Innovation & Product and Service Development at Turk Telekom.

Berk Canberk (Senior Member, IEEE) is a Professor within the School of Computing, Engineering and The Built Environment at Edinburgh Napier University, UK. He's also an Affiliated Professor within the Department of Artificial Intelligence and Data Engineering at Istanbul Technical University (ITU) and he is also the Innovation Director of BTS Group in Turkey.

Van-Linh Nguyen (Member, IEEE) is an Assistant Professor at the Department of Computer Science and Information Engineering, National Chung Cheng University (CCU), Taiwan.

Trung Q. Duong (Fellow, IEEE) is a Canada Excellence Research Chair (CERC) and a Full Professor at Memorial University of Newfoundland, Canada. He is also the adjunct Chair Professor in Telecommunications at Queen's University Belfast, UK and a Research Chair of Royal Academy of Engineering.