Empirically-Driven Adaptive Transmission for Enhanced Communication in Adversarial Environment

Emon Dey, Anuradha Ravi, Vinay Krishna Kumar, Jared Lewis, Nirmalya Roy
Center for Real-time Distributed Sensing and Autonomy (CARDS)
University of Maryland, Baltimore County, USA
{edey1, anuradha, vinayk2, jlewis9, nroy}@umbc.edu

***Abstract*—Efficient communication is vital for multi-agent robotic systems involving heterogeneous agents like Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) operating in dynamic and contested environments. The mobility of these agents, combined with varying environmental and channel conditions, often leads to increased latency and packet loss in communication between UAVs and UGVs. The disparity in velocity between these agents results in rapidly changing distances, which in turn affects key channel properties such as Received Signal Strength Indicator (RSSI), Signal-to-Noise Ratio (SNR), and the transition between line-of-sight (LOS) and non-line-of-sight (NLOS) conditions. In this paper, we investigate the impact of these channel parameters on inter-agent communication in operational settings and propose strategies to mitigate latency and packet loss from a transport layer perspective. While previous studies have explored methods to adjustable sliding window (ASW) sizes in the transport layer, they often lack the real-time adaptive capabilities needed to integrate physical layer properties for optimizing sliding window sizes. This study aims to fill these research gaps by introducing an innovative machine learning model that adaptively modifies the size of the sliding window in response to physical layer characteristics, including agent's velocity, data rate, distance, RSSI, and line-of-sight/non-line-of-sight scenarios. The efficacy of the proposed methodology is thoroughly assessed by co-simulations incorporating Gazebo for robot dynamics and NS-3 for network simulations. We effectively develop a middleware between Gazebo and NS-3 using the ROS framework and test our proposed methodology in contexts characterized by high mobility between the agents and intermittent connectivity caused by varying channel conditions. Our proposed methodology uses a state-of-the-art transformer-based DNN model that we train on our extensive dataset collected via simulation. The model takes in the various channel parameters and predicts the sliding window size for the given channel condition. The findings demonstrate substantial reductions by 10%, 12% in packet loss rates and communication latency when compared to static window size respectively.**

***Index Terms*—Machine Learning, Compression, NS-3, Gazebo, Co-simulation, Synchronization.**

I. Introduction

In the rapidly evolving field of autonomous robotics, efficient and reliable communication is a critical component, particularly in scenarios involving heterogeneous agents such as Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) [1]. These agents frequently encounter unstable communication links due to varying physical characteristics of the channel, which result in packet loss and network latency, thus affecting the system's performance, coordination, decision-making, and mission effectiveness. In the domain of multi-agent robotic systems, conventional methods for tackling communication obstacles usually rely on predetermined arrangements of network parameters [2].

Trivial solutions involve configuring fixed Transmission Control Protocol/Internet Protocol (TCP/IP) window sizes and adjusting data compression rates [3]. Nevertheless, these fixed configurations are unable to adjust to the constantly evolving conditions of the operational context, leading to less than ideal performance and ineffective allocation of resources. Researchers have suggested different adaptive communication solutions to address these difficulties, such as dynamic TCP window adjustment mechanisms, compression algorithms, and communication protocols that alter transmission rates based on the movement patterns of agents [4].

Although these efforts have demonstrated enhancements in specific situations, they frequently neglect the interaction between physical layer characteristics, communication settings, and network factors in diverse multi-agent robotic systems. That's why special focus is needed for resilient communication protocols that can adjust to the ever-changing and frequently unanticipated circumstances encountered by robotic systems in real-world scenarios. Unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs), operating in both line-of-sight (LOS) and non-line-of-sight (NLOS) environments, provide distinct issues in terms of ensuring consistent and efficient data transmission. Conventional TCP/IP mechanisms [5] are not fundamentally equipped to handle the swift alterations in network structure and data flow demands that are common in mobile robotic agents. As a consequence, there is an increase in packet loss and latency, which directly affects operational efficiency and mission success.

The objective of this study is to utilize machine learning methods to dynamically modify TCP/IP windows. This will result in the development of a communication framework that can effectively satisfy the requirements of existing robotic systems and establish a basis for future developments. Although there has been extensive study on enhancing TCP/IP performance [6], current methods frequently concentrate on uniform environments, single-variable adjustments [7], or necessitate

substantial processing resources [8]. This paper presents a novel machine learning-driven approach to dynamically adjust the TCP/IP window size based on key physical layer properties such as agent velocities, inter-agent distances, communication scenarios (line-of-sight or non-line-of-sight), and data rates. By intelligently selecting an appropriate window size that matches the operating conditions, our method aims to mitigate packet loss stemming from velocity disparities among robots like unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs). The major contributions we claim here are:

- *Developed a machine learning model to dynamically adjust TCP/IP window based on physical layer properties* We design an adaptive and intelligent selection process of TCP/IP window (ASW) based on the participating agents velocities, distance, communication scenario, RSSI value, and data rate. This helps to address the packet loss and latency due to higher velocity disparity among the robotics agents i.e., UAVs and UGVs and also setting up appropriate packet volume to be sent based the communication environment.
- *Conducted comprehensive empirical analysis to better understand the sim-to-real transfer scenario* We present our experiment, taking into account different communication scenarios on the basis of probability of packet loss, and average packet delay, and compression rate achieved. We have employed Gazebo as Physics simulator and NS-3 as network simulator to procure a comprehensive simulation data for training and also to report our experimental results. Results show that our approach works better in comparison with the traditional fixed window based method in ensuring fewer packet losses and reducing the transmission latency even in the existence of heterogeneous agents.

II. Related work

In this section, we will briefly outline the work that has been done in relation to the various aspects of our system.

A. Machine Learning in Communication

Machine learning techniques have become highly effective instruments for improving and optimizing different facets of communication networks. In order to overcome obstacles in fields like channel coding, estimation, signal identification, and resource allocation, researchers and engineers have created creative solutions by utilizing machine learning algorithms' capacity to learn from data and spot patterns [9]. For example, deep learning has been investigated recently for efficient channel code creation [10] and for joint channel estimation and signal detection. In wireless networks, machine learning has also been used for resource allocation tasks including spectrum management and power control [11]. One well-known use is the dynamic adjustment of network settings, such data compression rates and TCP window sizes, to correspond with the constantly shifting circumstances of the communication environment [12]. The best settings for these parameters can be predicted by machine learning models, which can be trained on past data or in-the-moment observations. This ensures effective resource use, minimizes packet loss, and lowers network latency [13].

B. Synchronization in Simulation

While a comprehensive assessment of currently utilized simulators is beyond the scope of this paper, we highlight a few recent noteworthy physics and network simulators that are most relevant to our setting and have had a significant influence on this study. CORNET 2.0 [14] extended the implementation of CORNET to make it applicable to any robotic framework and the scalability to manage an increasing number of robots has been demonstrated. FlyNetSim [15] maintains a time-stepped-schedule mechanism, however; simulated network events must be buffered until the next sample time if a network simulator runs faster than the physical simulation. ROS-NetSim [16] is a sliding window based technique that keep track of and record network events during the course of the window period, and then enable the network simulator to step up to and through the end of the window, but it is difficult to configure the appropriate window size for multi-agent systems. In this consideration, we devised the sliding window method for our heterogeneous multi-agent system.

III. Methodology

This workflow of our integrated simulation framework where we illustrate the interaction among the application model, network model, and channel model (fig. 1). The application model is designed with a Physics simulator. The Network Model operates within a network simulator, enhanced by a multi-head ML model for functionalities integrated with an adaptive sliding window at the transport layer. The network model simulates WiFi communications and the Internet, while the channel model considers different communication scenarios, impacting the movement of different entities. Here, we explain the formulation of the ML model, and synchronization between the two simulators.

A. Designing Machine Learning based sliding window keeping physical layer properties in the loop

To predict the suitable sliding window size (W) to minimize packet loss and to optimize latency while ensuring data fits within the available bandwidth, we designed our proposed machine learning model which will account for the transport layer (sliding window size).

Notations and Inputs

- V_i: Velocity of the robotic agents (meters per second)
- d: Distance between agents (meters)
- R_s: Received Signal Strength Indicator (dB)
- D_r: Data rate (bits per second, bps)
- W: Sliding window size (number of packets)
- L_p: Packet loss (fraction)
- L_{pT}: Packet loss threshold (fraction)
- T_p: Latency (milliseconds, ms)
- T_{pT}: Latency threshold (milliseconds, ms)

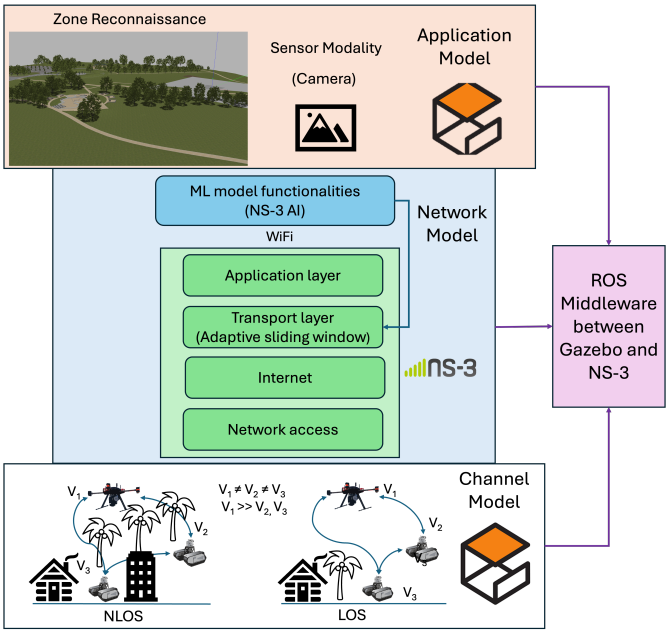

Fig. 1: Modeling of proposed ML-driven network topology and co-simulation framework.

Objective

Minimize packet loss (L_p) and (T_p) by predicting the optimal sliding window size (W).

Machine Learning Model Development We develop a multi-head Multi-Layer Perceptron (MLP) to predict both W and C based on the input features V_i, d_i, D_{ri}, R_{si}. Let:

$$\mathbf{X} = [V, d, D_r, R_s]$$

be the input features vector.

$$W = f_W(\mathbf{X}; \theta_W)$$

is the machine learning model predicting the sliding window size, where θ_W represents the parameters of the model.

$$C = f_C(\mathbf{X}; \theta_C)$$

is the machine learning model predicting the compression ratio, where θ_C represents the parameters of the model.

Loss Function for Training the Model To train the model, we need two loss functions: one for minimizing packet loss and another for optimizing latency. The total loss function will be a weighted sum of these two losses.

Packet Loss Prediction We have collected a dataset with observed packet loss for different window sizes under various network conditions, we can train the model to minimize the packet loss. The reference packet loss values are generated using the following equation: Packet loss probability,

$$^{i}_{n}Lp = 1 - \frac{^{i}_{n}D_s}{^{i}_{n}D_p} \tag{1}$$

Here, $^{i}_{n}D_s$ is the number of data packets delivered to the total number of subscribers and $^{i}_{n}D_p$ is the number of data packets transmitted from the publishers.

Latency Prediction Similarly, we procured a dataset with observed latency for different compression ratios under various network conditions. While recording the latency values we considered the followinf parameters: Average packet delay,

$$T_p = T_{pr} + T_t + T_{pg} + T_q \tag{2}$$

Here, T_{pr} is processing delay T_t is transmission Delay, and T_{pg}, T_q represent propagation, and queuing delay respectively.

Training Dataset

Predictor variables = $(V_i, d_i, D_{ri}, R_{si})$

Target variable = W_i

Response variables = (L_{pi}, T_{pi})

represents the observed agents velocity, distance, data rate, window sizes, signal strength, packet loss, and latency for sample i.

Model Training We minimize the combined loss function:

$$\mathcal{L}(\theta_W) = \alpha\mathcal{L}_P(\theta_W) + \beta\mathcal{L}_L(\theta_w)$$

where:

$$\mathcal{L}_P(\theta_W) = \frac{1}{n}\sum_{i=1}^{n}(L_{p,i} - \hat{L}_{p,i})^2$$

$$\mathcal{L}_L(\theta_w) = \frac{1}{n}\sum_{i=1}^{n}(T_{p,i} - \hat{T}_{p,i})^2$$

α and β are weighting factors that balance the importance of packet loss and latency. $\hat{L}_{p,i}$ is the predicted packet loss given the network conditions and predicted window size. $\hat{T}_{p,i}$ is the predicted latency given the network conditions.

Packet Loss Prediction Model

Let $g(V, d, D_r, R_s, W)$ be a function that models the packet loss based on the network conditions and window size.

Latency Prediction Model Let $h(V, d, D_r, R_s, W)$ be a function that models the latency based on the network conditions and window size.

Combined Optimization The combined approach can be formulated as:

$$\min_{\theta_W}\left(\alpha\frac{1}{n}\sum_{i=1}^{n}\left(P_{L,i} - g(f_W(\mathbf{X}_i;\theta_W))\right)^2 + \beta\frac{1}{n}\sum_{i=1}^{n}\left(L_i - h(f_w(\mathbf{X}_i;\theta_w))\right)^2\right)$$

B. Addressing synchronization issue during simulation

Another essential part of our process is the development of a mutual information update method for both the physics simulator and the network simulator. At the outset of the simulation, the physics simulator establishes two essential characteristics of the agents that will be employed for a particular communication round. These characteristics are the velocity of the agents and the distance that separates them as determined by positional coordinates. The information that the network simulator stores includes the communication scheme that will be usedthe number of packets, the length of each packet, and the IP addresses of both the publisher and subscriber agents 1.

After that, the physics simulator and the network simulator are iterated upon while using a single initially fixed window size. When the simulation is advanced, the physics simulator sends the information about distance and velocity to the network simulator, and the network simulator uses that

information to compute the chance of packet loss. When the value of the loss is compared to a predetermined threshold, the result is sent to the display. In the event that this is not the case, the information is sent back to the synchronizing module, where it is processed, and the initial window size is modified in accordance with the method 1. This process will continue until a result is obtained for this particular round that is deemed to be satisfactory, after which the initialization procedure will begin once again with a fresh group of agents.

Algorithm 1 Pseudo-code for adaptive sliding window (ASW) size determination

Input: W_0, V, d, D_r, R_s, L_{pT}, T_{pT}
Output: L_p, T_p
Simulation Initialization:
$t \leftarrow 0$
$W \leftarrow W_0$ ▷ Initialize with default window size
$\mathbf{X} \leftarrow [V, d, D_r, R_s]$ ▷ Feature vector
$[W_i] \leftarrow$ ML Model$(\mathbf{X})$ ▷ Predict initial window size
Data Transmission and Update:
Transmit data packets and monitor conditions
Adaptation:
while network event occurs **do**
    $L_p \leftarrow f_L(D_r, W, V, d, R_s)$
    $T_p \leftarrow f_T(D_r, W, V, d, R_s)$
    if $L_p > L_{pT}$ **and** $T_p > T_{pT}$ **then**
        $[W_{\text{new}}] \leftarrow$ ML Model$(\mathbf{X})$ ▷ Predict new window
    else
        $W \leftarrow W_{\text{new}}$ ▷ Update window size
    end if
end while
Report T_p **and** L_p **upon synchronization:** Calculate average delay and packet loss probability for the synchronized event using equations 2 and 1 respectively
Timestamp update: *Update* $t = t + W$ and request for next window to iterate the process

IV. Experimental Details

A. Simulation Setup

We have collected data from the simulation environment where we have arranged a specific co-simulation setup using Gazebo as physics simulator, ns-3 as network simulation and a customized ROS-based middleware to synchronize those the twos simulators with different operating principle. As robotic agents we have couple of simulated Clearpath Jackal as UGVs and one IRIS drone as UAV. We have used the image data streamed from the integrated camera of those robotics agents as the data modality to be sent across agents. The leverage that we got with this specific setup is that we simulated different LOS and NLOS scenarios, velocity variation, distance variation, air-ground communication, which aids us in developing a comprehensive dataset. The features we have in our dataset are velocity of the UAV/UGV, distance, RSSI value, data rate, and sliding window size for Wi-Fi. We measure the packet loss and delay for different scenarios. The goal of the optimization function is to predict the appropriate sliding window size for a given channel condition which will give lowest packet loss and latency values. The threshold values of packet loss percentage (L_{pT}) and latency (T_{pT}) values we have set for our specific use case are '60%' and '2000mS' respectively.

The proposed machine learning model was linked with the simulation system to optimize data transmission through the use of sliding window arrangement. The procedure starts with the network nodes being initialized. Next, the physical layer is simulated, and important characteristics like RSSI, data rate, and mobility are modeled to mimic real-world communication settings. Data routing between nodes is managed and IP addresses are assigned via the TCP/IP stack. The data is subsequently subjected to transport layer which improves efficiency by modifying sliding window sizes in response to network conditions. The foundation of the system is the proposed machine learning model, which evaluates network performance and communicates with the window adjusting module to make real-time modifications that enhance data transfer in general.

B. Details on the Machine Learning Model

We have used the pretrained weights of the TabNet [17] model and performed a supervised fine-tuning on top of it for our use-case. TabNet is on Transformer architecture for tabular data employs a sequential attention mechanism which provides better interpretability. Further Sparsemax mechanism is used instead of Softmax which uses a subset of features chosen at each decision step. The model is re-trained for 150 epochs with adjust weight decay enabled and 'RMSE' is used as the performance metric. The train-split of the dataset contained specific network scenarios and while testing we enforced similar but unique cases. The ground truth for the

Velocity_diff (m/S)	Distance (m)	Data_rate (Kbps)	RSSI	Sliding window (Bytes)	Packet loss (%)	Latency (mS)
6	180	471.42	-45	344	38	392.46
8	115	128.76	-32	784	26	222.24
12	230	112.45	-62	512	55	594.88

TABLE I: Training data samples.

sliding window size are generated based on the following assumptions while generating the dataset:

- The propagation delay increases in direct proportion to the distance between two points.
- The packet loss probability is assumed to be directly affected by the Received Signal Strength Indicator (RSSI).
- The exponential effect of Received Signal Strength Indicator (RSSI) on the sliding window.
- Signal quality can deteriorate and the probability of errors or retransmissions can increase with larger distances and higher relative velocities.
- The multipliers for each parameter are derived by empirical methods in order to preserve the relative importance of the parameters.

We start with a pre-calculated default window size based on those assumptions and we vary the interval by 8 bytes to find the five different neighbouring window sizes. Among those, the one for which we find the lowest packet loss and latency,

we mark that window as ground truth. We have included some training data samples in table I generated which are used to develop the ML model. While collecting the data we kept the velocity difference and distance constant for different iterations and varied the other network parameters to record the latency and packet loss.

V. Result Analysis

A. Parameter Selection

In this section we analyze the characteristics that significantly affect latency and packet loss, as indicated in fig. 2. We provide justification of choosing data Rate, distance, RSSI, and sliding window as key parameters to understand the network performance and enhancing it by dynamically modifying the sliding window size.

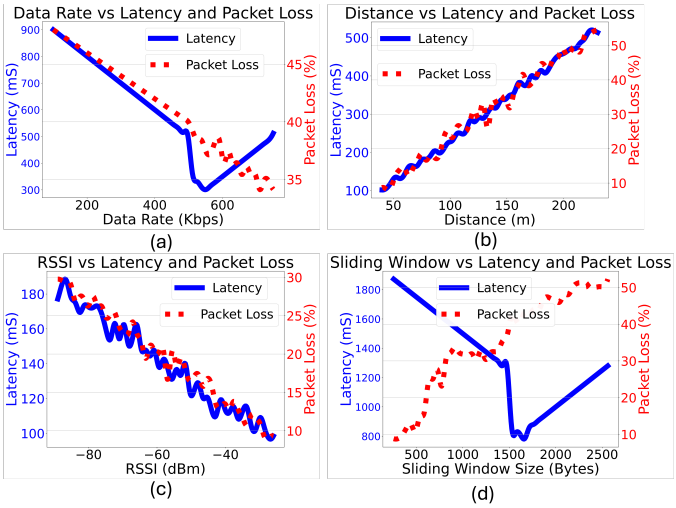

Fig. 2: Latency and packet loss trend based on individual predictors.

As shown in plot 'a' of fig. 2 initial data rate increases reduce delay, indicating more efficient transmission. Beyond a certain point, increasing data rate sharply increases packet loss and latency. At larger data rates, the network gets crowded, causing re-transmissions and increasing latency. Also, transmitter-receiver distance strongly affects latency and packet loss. Latency and packet loss increase gradually with distance (plot 'b'). Distance usually weakens signals and increases error risk, requiring additional re-transmissions. Adjusting the sliding window size based on distance can help alleviate these impacts by allowing delays without increasing network congestion. Another parameter we investigated is RSSI value where we can notice the inverse relationship between latency and packet loss with RSSI (plot 'c'). This emphasizes signal strength's relevance in a reliable and efficient connection. Stronger signals reduce mistakes and packet loss, lowering latency. In settings with fluctuating signal strength, RSSI-based sliding window size modification can optimize performance. The last plot ('d') shows how sliding window size affects network performance. Latency and packet loss rise with sliding window size but stabilize or decrease. A bigger window size can increase latency due to acknowledgment delays, but it can reduce packet loss by enabling more packets to be in transit. Beyond a threshold, expanding the window size may reduce returns or degrade performance owing to network congestion. To optimize latency and packet loss, the sliding window size must be constantly adjusted based on network conditions.

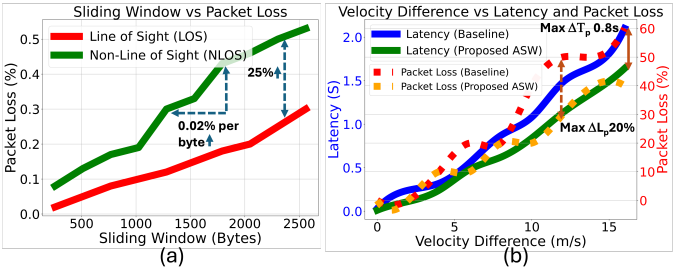

Fig. 3: (a) Performance after applying proposed ML-based adaptive transmission, (b) Comparative results between the baseline fixed window and compression ratio and our proposed adaptive sliding window (ASW) based transmission strategy.

B. Prediction Performance and Baseline Comparison

We have experimented with extensive communication scenarios varying the network parameters to showcase the efficacy of our proposed model. Firstly, to understand how our proposed ML model is performing in terms of packet loss percentage while predicting the appropriate sliding window in two distinct scenarios: line-of-sight (LOS) and non-line-of-sight (NLOS) circumstances. The findings are illustrated in the fig. 3. To measure the performance of our adaptive transmission in terms of sliding window size and respective packet loss values, we vary the sliding window size is from 256 bytes to 2560 bytes. At a sliding window size of 500 bytes, for example, the packet loss is negligible for both cases; but, as the window size increases, the difference between LOS and NLOS packet loss widens dramatically. By the time the sliding window reaches 2500 bytes, the packet loss caused by NLOS is greater than 0.5%, but the loss of packets caused by LOS is somewhat less than 0.3%.

It can be deduced from this that bigger sliding window sizes result in higher packet losses, particularly in non-line-of-sight (NLOS) conditions. This is because of the increased chance of interference and signal degradation that occurs in such circumstances. The visualization of the baseline comparison can be found in fig. 3. This graph shows how the change in speed (in meters per second) affects latency (in seconds) and packet loss (in percentage) for both the proposed system (ASW) and the baseline model with fixed window and compression ratio. For the baseline and proposed systems, the blue and green lines show the delay. For the baseline and proposed systems, the red and yellow dashed lines show the packet loss. In both systems, latency and packet loss go up as the velocity gap grows. The proposed system, on the other hand, constantly has less latency and packet loss than the baseline. When the speed difference is about 12.5 m/s, the biggest difference in latency between the standard system and the proposed system

is about 0.8 seconds. Our proposed system also has a better performance in high-speed situations, as shown by the fact that the maximum reduction in packet loss is about 20%. We have also listed five different test case scenarios where the ML model has predicted the sliding window and we measure the packet loss percentage and latency for those cases. The results can be found in table II and while comparing training samples in table I we can find the the trend is learnable and the predicted sliding size can actually impact lower latency and packet loss percentage.

Velocity_diff (m/S)	Distance (m)	Data_rate (Kbps)	RSSI	Sliding window (Bytes)	Packet loss (%)	Latency (mS)
0	175	499.40	-64	368	34	453.78
5	210	642.87	-74	496	56	258.97
7	147	300.09	-52	680	29	312.05
10	172	112.68	-56	416	34	382.38
12	130	120.60	-43	712	25	408.12

TABLE II: Predicted sliding window values for different sets of network parameters.

VI. Discussions and Future Direction

Real-world performance emulation using time-based network simulators. In this work, we employ the event-driven network simulator and we have used a synchronizing middleware to continue the co-simulation with the Physics simulator. As integration of most of the prominent time-based network simulator i.e., EMANE with ROS is a separate and still ongoing research, this is beyond the scope of this study. But it would be worth investigating whether replacing the event-based network with time-based can model the complicated interactions better in mobile and wireless networks, where latency, jitters, and throughput change over time.

Enhancing the ML models capability by employing automated communication scenario (LOS/NLOS) detection. Using ML models to automatically recognize to LOS and NLOS communication circumstances could be an interesting extension of this study for better network optimization and problem prediction. Existing research in this domain has shown through automated detection of certain scenarios, the real-time alteration of OSI layer parameters can be done more effectively and the latency due to the parameter selection essentially can be reduce. We are actively investigating to incorporate advanced signal processing and real-time data into our proposed ML model's workflow to develop more robust modelling of the communication scenario.

VII. Conclusion

In this work, we have presented the overall design of a machine learning model-driven intelligent selection of the sliding window size of a TCP/IP network, which will take into account the physical layer parameter values during the transmission. Our approach showed a significant improvement in understanding the communication scenario to setup the packet volume to be transferred, resulting in a lower packet loss. Additionally, our proposed an adaptive sliding window (ASW) approach reduced the communication overhead by a significant margin, reducing the overall transmission latency. Also, we address the synchronization issue between physics-based and network simulators while simulating the performance of our proposed method. Furthermore, we have generated a comprehensive dataset simulating diverse communication scenarios to accurately model the real-world intricacies and performed extensive analysis to minimize the sim-to-real gap.

Acknowledgment

This work has been supported by NSF grant #2233879.

References

[1] Byungjun Kim, Christoph Mecklenbräuker, and Peter Gerstoft. Deep learning-based modulation classification of practical ofdm signals for spectrum sensing. *arXiv preprint arXiv:2403.19292*, 2024.

[2] Nariman Farsad and Andrea Goldsmith. Neural network detection of data sequences in communication systems. *IEEE Transactions on Signal Processing*, 66(21):5663–5678, 2018.

[3] Hamidah Alanazi, Shengping Bi, Tao Wang, and Tao Hou. Adaptive feature engineering via attention-based lstm towards high performance reconnaissance attack detection. In *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*, pages 542–547. IEEE, 2023.

[4] Andreas Andersson, Kristoffer Hägglund, and Erik Axell. Deep learning-based demodulation in impulse noise channels. In *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*, pages 574–579. IEEE, 2023.

[5] Jiacheng Xia, Gaoxiong Zeng, Junxue Zhang, Weiyan Wang, Wei Bai, Junchen Jiang, and Kai Chen. Rethinking transport layer design for distributed machine learning. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking*, pages 22–28, 2019.

[6] Lavy Libman and Ariel Orda. Optimal sliding-window strategies in networks with long round-trip delays. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 3, pages 2057–2067. IEEE, 2003.

[7] Ernestina Cianca, Aldo De Luise, Marina Ruggieri, and Ramjee Prasad. Channel-adaptive techniques in wireless communications: an overview. *Wireless communications and mobile computing*, 2(8):799–813, 2002.

[8] Zayan El Khaled, Hamid Mcheick, and Fabio Petrillo. Wifi coverage range characterization for smart space applications. In *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, pages 61–68. IEEE, 2019.

[9] Afsaneh Mahmoudi, Hossein S Ghadikolaei, and Carlo Fischione. Cost-efficient distributed optimization in machine learning over wireless networks. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2020.

[10] Qian Huang and Zhimin Tang. High-performance and lightweight ai model for robot vacuum cleaners with low bitwidth strong non-uniform quantization. *AI*, 4(3):531–550, 2023.

[11] Yu Jin, Jiayi Zhang, Shi Jin, and Bo Ai. Channel estimation for cell-free mmwave massive mimo through deep learning. *IEEE Transactions on Vehicular Technology*, 68(10):10325–10329, 2019.

[12] Emon Dey, Jumman Hossain, Nirmalya Roy, and Carl Busart. Synchrosim: An integrated co-simulation middleware for heterogeneous multi-robot system. In *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 334–341. IEEE, 2022.

[13] Qiang Hu, Feifei Gao, Hao Zhang, Shi Jin, and Geoffrey Ye Li. Deep learning for channel estimation: Interpretation, performance, and comparison. *IEEE Transactions on Wireless Communications*, 20(4):2398–2412, 2021.

[14] Srikrishna Acharya, Amrutur Bharadwaj, Bharatheesha Mukunda, and Yogesh Simmhan. Cornet 2.0: A co-simulation middleware for robot networks. ArXiv, 2021.

[15] Sabur Baidya, Zoheb Shaikh, and Marco Levorato. Flynetsim: An open source synchronized uav network simulator based on ns-3 and ardupilot. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 37–45, 2018.

[16] Miguel Calvo-Fullana, Daniel Mox, Alexander Pyattaev, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. Ros-netsim: A framework for the integration of robotic and network simulators. *IEEE Robotics and Automation Letters*, 6(2):1120–1127, 2021.

[17] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687, 2021.