




Article

Intelligent Packet Priority Module for a Network of Unmanned Aerial Vehicles Using Manhattan Long Short-Term Memory

Dino Budi Prakoso¹, Jauzak Hussaini Windiatmaja¹ , Agus Mulyanto¹, Riri Fitri Sari^{1,*} 
and Rosdiadee Nordin^{2,*} 

¹ Department of Electrical Engineering, University of Indonesia, Depok 16424, Indonesia; dino.budi@ui.ac.id (D.B.P.); jauzak.hussaini@ui.ac.id (J.H.W.); agus.mulyanto@ui.ac.id (A.M.)

² Department of Engineering, School of Engineering and Technology, Sunway University, 5, Jalan Universiti, Bandar Sunway 47500, Selangor, Malaysia

* Correspondence: riri@ui.ac.id (R.F.S.); rosdiadeen@sunway.edu.my (R.N.)

Abstract: Unmanned aerial vehicles (UAVs) are becoming more common in wireless communication networks. Using UAVs can lead to network problems. An issue arises when the UAVs function in a network-access-limited environment with nodes causing interference. This issue could potentially hinder UAV network connectivity. This paper introduces an intelligent packet priority module (IPPM) to minimize network latency. This study analyzed Network Simulator-3 (NS-3) network modules utilizing Manhattan long short-term memory (MaLSTM) for packet classification of critical UAV, ground control station (GCS), or interfering nodes. To minimize network latency and packet delivery ratio (PDR) issues caused by interfering nodes, packets from prioritized nodes are transmitted first. Simulation results and evaluation show that our proposed intelligent packet priority module (IPPM) method outperformed previous approaches. The proposed IPPM based on MaLSTM implementation for the priority packet module led to a lower network delay and a higher packet delivery ratio. The performance of the IPPM averaged 62.2 ms network delay and 0.97 packet delivery ratio (PDR). The MaLSTM peaked at 97.5% accuracy. Upon further evaluation, the stability of LSTM Siamese models was observed to be consistent across diverse similarity functions, including cosine and Euclidean distances.

Keywords: UAV communications; FANET; FlyNetSim; NS-3; MaLSTM; packet priority



Citation: Prakoso, D.B.; Windiatmaja, J.H.; Mulyanto, A.; Sari, R.F.; Nordin, R. Intelligent Packet Priority Module for a Network of Unmanned Aerial Vehicles Using Manhattan Long Short-Term Memory. *Drones* **2024**, *8*, 183. <https://doi.org/10.3390/drones8050183>

Academic Editors: Chinthaka Premachandra and Tomotaka Kimura

Received: 13 March 2024

Revised: 4 May 2024

Accepted: 4 May 2024

Published: 7 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

An unmanned aerial vehicle (UAV) is a type of remote-controlled airplane. UAVs have the option of being remotely piloted in real time or flying independently along pre-determined routes. This variety of aircraft, known as drones, is on the rise across all industries. The increased use of UAVs in daily life has piqued the interest of researchers who study UAVs. Currently, 68% of UAV purchases and usage are for professional purposes, according to a report by Skylogic Research [1]. UAVs are utilized for a variety of professional applications, including agricultural surveillance, disaster management, and environmental monitoring [2,3]. UAVs find application across diverse industries, encompassing agriculture, disaster relief, and environmental research, among others.

The UAV communication infrastructure can function as an extended wireless communication range connection [4]. Additionally, Callegro et al. [5,6] investigated UAVs and networks. Real-world UAV networks face numerous logistical obstacles, particularly in urban or densely populated areas. Due to the nature of UAVs, a large operating area is required. Therefore, digital simulation is extremely essential for UAV and network research.

FlyNetSim is a prominent example of a currently available simulator for UAV networks. The network simulation in FlyNetSim is implemented using Ns-3, while the configuration of UAVs is facilitated via ArduPilot. NS-3 is an open-source network simulator that is

supported by a community of developers and users [7]. ArduPilot is a simulator that simulates UAVs with capabilities for control, navigation, and route planning. ArduPilot is also capable of controlling submarines, ships, rovers, multi-copters, helicopters, and fixed-wing aircraft [8]. FlyNetSim can simulate the interactions between UAV operation and network environment, including sensing, navigation, and battery health [9].

The utilization of UAVs may encounter challenges associated with network connectivity. An illustrative instance arises when the UAV is situated in a geographical area characterized by the presence of multiple interfering nodes vying for network channels. Interference in communication networks is a pervasive challenge that can affect the performance and reliability of UAV communication systems, regardless of the underlying network technology, including 4G, 5G, and beyond. In areas where UAVs operate, such as urban environments or airspace shared with other wireless devices, frequency congestion can occur. This congestion can lead to interference as multiple devices compete for limited frequency bands, resulting in degraded signal quality and increased packet loss. This matter has the potential to lead to delays in network connectivity. The latency of the unmanned aerial vehicle (UAV) experiences an increase in proportion to the number of nodes that are causing interference.

To remedy the latency issue, Jauzak et al. [10] devised a socket priority module. This socket priority module assigns transmissions from UAVs and their GCS with priority identifiers. Nonetheless, UAV applications may experience network-related issues. One of the factors contributing to the challenges faced by UAVs is the presence of numerous interfering networks in their operating area, resulting in contention for network channels. This phenomenon has the potential to result in delays inside the network. As the number of interfering nodes increases, the delays experienced by the UAV is proportionally amplified. We constructed an intelligent package priority module to determine the priority of packets that the network processes, in order to reduce delays and increase the PDR in wireless communications caused by interfering nodes. The specific contributions made by this endeavor include the following:

- (1) Design and implementation of an IPPM using a deep learning algorithm, MaLSTM, in a NS-3 environment with a Python binding module;
- (2) Via the proposed IPPM, provision of an efficient solution to a number of issues induced by interfering nodes, related to minimizing delays and increasing PDR in wireless communication between UAVs and their GCS.

This paper is organized in the following manner. The related work and current state of the art in this domain are discussed in Section 2. Section 3 presents a comprehensive overview of the intelligent packet priority module that is proposed. The performance evaluation of our proposed method is presented in Section 4, including the results, discussions, and analysis. The concluding section of the paper is denoted as Section 5.

2. Related Work

2.1. UAV

The main goal of UAV models is to mimic how UAV systems work and how they move through the air. Some models, like the H-SIM Flight Simulator [11], can simulate planes that are autonomously driven. PX4 and ArduPilot are two software applications commonly employed by individuals to operate and manage diverse setups of UAVs, encompassing multirotors, fixed-wing aircraft, and helicopters, as well as ships and rovers. Also, compared with PX4, it supports a wider range of systems, such as Erle-Brain [12], Pixhawk [13], and NAVIO [14].

There are five main components that make up ArduPilot: the vehicle code, the tool files, the open-source library, the hardware abstraction layer (AP HAL), and the external support code (such as MAVLink and DroneKit) [15]. Figure 1 shows how the main parts of ArduPilot are put together. The ArduPilot software platform facilitates the development of specialized UAV simulators to augment their operational capabilities.

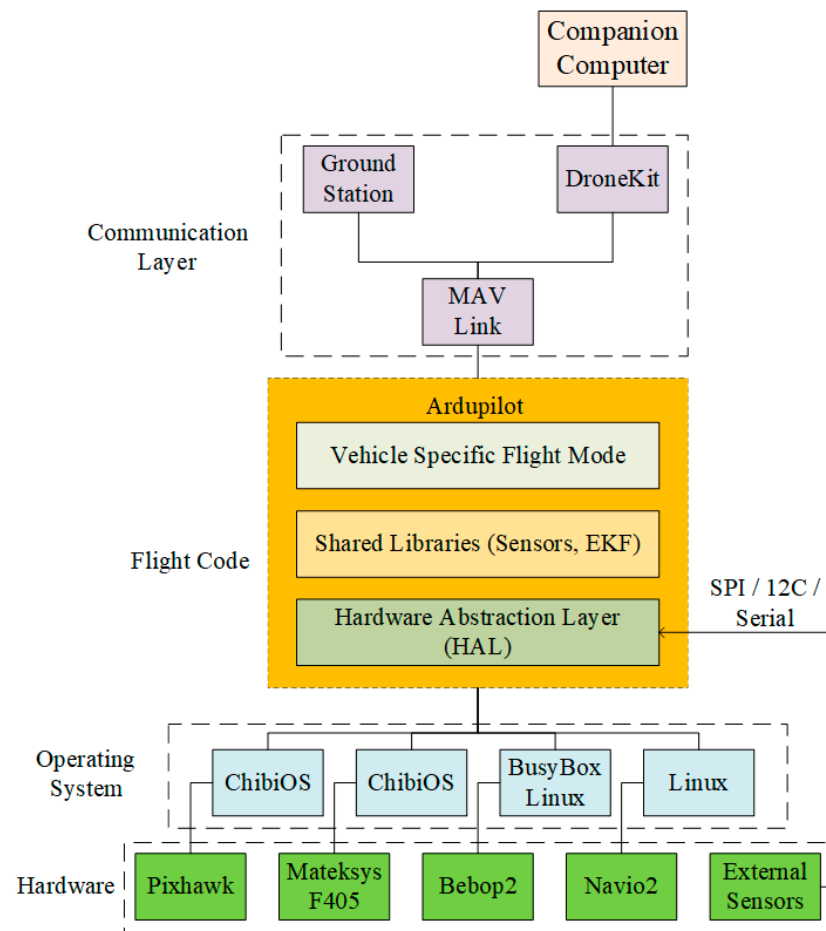


Figure 1. Ardupilot Structure.

The robot operating system (ROS) [16] helps Gazebo [17] simulate aircraft, land rovers, and miniature robots. The Gazebo simulator's size makes simulating multiple vehicles in one environment difficult. XPlane-10 and RealFlight use ArduPilot. Most of the models address navigation and mobility issues. FlyNetSim needs software in the loop (SITL) for autonomous navigation, especially ArduPilot software. ArduPilot runs quickly on end-user PCs without hardware, thanks to SITL. The approach uses ArduPilot's ability to automate across platforms. SITL uses sensor data from a flight simulator's flight dynamics model. The ArduPilot system can connect multiple vehicle simulators and external simulators. SITL lets ArduPilot use multi-rotor aircraft, antenna trackers, camera gimbals, underwater vehicles, fixed-wing aircraft, ground vehicles, lidar, and optical flow sensors.

2.2. Flynet Simulator

FlyNetSim is an open-source simulator that model networks and UAVs. It was developed using NS-3 and ArduPilot. The implementation of FlyNetSim holds promise in augmenting the experimental procedure by means of simulating an edge server, a UAV, and wireless networks. In contrast to alternative integrated UAV-network simulators, FlyNetSim possesses the capability to effectively manage diverse UAV-control scenarios, encompassing the transmission of data streams, the regulation of telemetry, and the utilization of multiple UAVs.

The architecture of FlyNetSim is illustrated in Figure 2, comprising four main components:

1. A fly simulator facilitates the launch and operation of UAVs, GCS, and visualization tools;
2. A network simulator replicates the real-world network architecture and its surrounding nodes that compete with each other;

3. Middleware enables bidirectional network communications based on ZeroMQ;
4. Autogen is proposed for the purpose of creating a comprehensive map of the various components of the UAV, including sensors, as well as their corresponding network nodes.

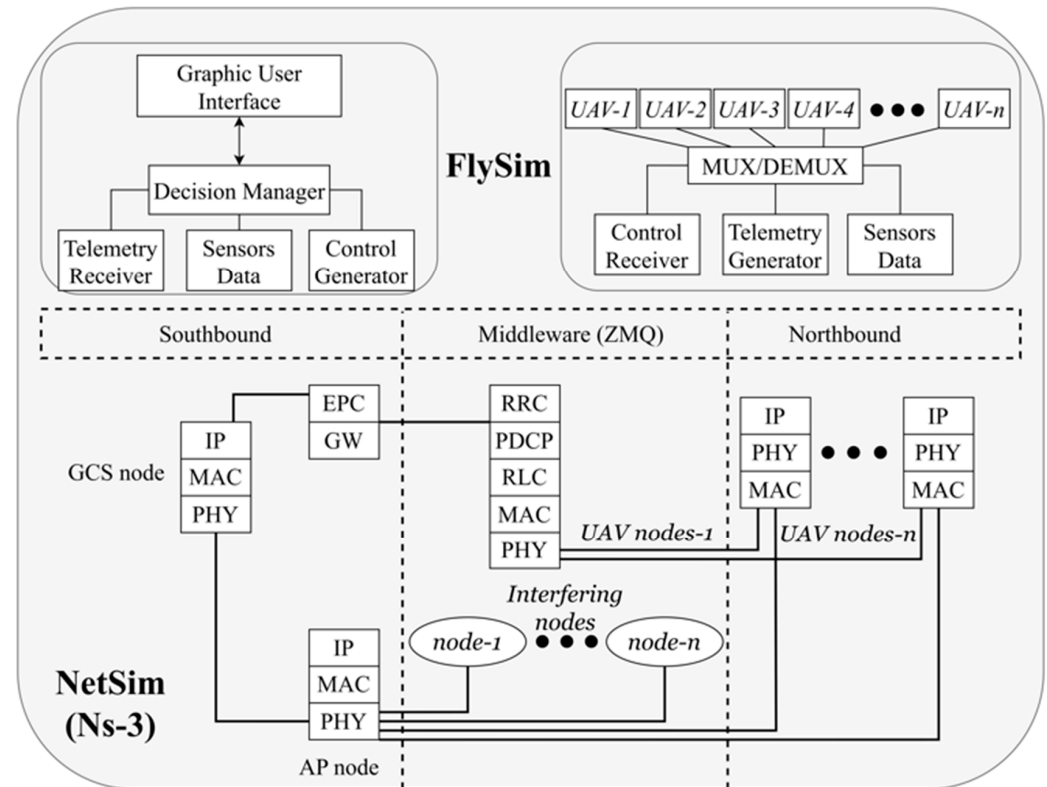


Figure 2. FlyNetSim Architecture.

2.3. Machine Learning for Priority-Related Module

Advances in artificial intelligence (AI), specifically in machine learning (ML) and deep learning (DL), have led to significant accomplishments across various disciplines. Deep learning algorithms are considered to be one of the most impactful and innovative paradigms for wireless radio technology in upcoming network generations. It is anticipated that forthcoming wireless networks will facilitate exceptionally elevated rates of data processing and innovative applications, thereby mandating radio support for intelligent adaptive learning and decision making. In recent articles, authors have highlighted the most impressive achievements of machine learning techniques in comparison to conventional methods and have proposed research recommendations on networking utilizing ML or DL to motivate academicians to develop innovative algorithms. Researchers can use ML/DL to design and optimize UAV-based wireless communication systems [18].

Several studies have provided summaries of the application of ML/DL to wireless communication problems. Luong et al. [19] provided an exhaustive overview of deep learning for applications including data offloading, wireless caching, data rate control, network security, dynamic network access, and connectivity preservation. Other studies have examined the implementation of deep learning algorithms in particular domains, such as mobile edge caching [20], vehicular networks in 6G [21], URLLC in 6G networks [22], and the Internet of Things (IoT) [23]. Interference from unauthenticated nodes contributes to high network latency and low PDR in the communication between UAV and GCS. For this reason, QoS must be implemented for managing packet delivery. Bezmenov et al. [1] applied the reinforcement learning (RL) algorithm to predictive quality of service (PQoS) to optimize the efficiency and adaptability of wireless network communication

through accurate predictions. In addition, for the UAV and GCS interfaces, Jauzak et al. [10] proposed a module priority setting. The objective of this prioritization is to give precedence to drones and GCS packets. In order to mitigate the latency resulting from the presence of interfering nodes, the payload with the highest priority is ultimately transmitted first. That study was carried out by implementing a prioritization mechanism that assigned a higher priority to sockets originating from trusted nodes in order to enhance network performance. This present investigation does not incorporate a deep learning/machine learning strategy.

Jeganathan et al. [24] proposed a UAV-assisted, cooperative wireless communication system to transmit sensor observations to a terrestrial base station. The average age of the information was predicted using LSTM algorithms. The age of information (AoI) concept combined with full-duplex unmanned aerial vehicles (FD-UAVs) can improve wireless-powered sensor network latency and spectrum efficiency. AoI reduced latency through identifying system parameters that cause it, while maintaining QoS. The results suggested a positive correlation between self-interference (SI) and average interference amplitude (AAoI). To determine how design parameters affect wireless systems used by UAVs, a comprehensive analysis is needed [18]. UAV systems must be evaluated using QoS metrics to determine their effectiveness. Coverage probability, throughput, latency, and reliability are important metrics. Performance evaluations can also reveal UAV design tradeoffs.

Umair et al. [25] proposed a maximum entropy classifier and multilayer deep neural network (DNN) for ITC. The Moore dataset was initially processed using OneHotEncoding for categorical variables. After preprocessing, the data were split into training and testing sets. An extra-trees classifier selected common flow features from preprocessed data. After training the feedforward DNN, the output layer used a maximum entropy classifier. The proposed multilayer DNN had 99.23% accuracy. Several SVM and KNN tests were run for comparison. These tests showed SVM accuracy of 98.90% and KNN accuracy of 98.56%. Multilayer DNN has great potential, as shown by that comparison. Khawaja et al. [26] proposed using multi-agent reinforcement learning (RL) with MMPAR on UAVs to detect, track, and classify UAVs in swarms. Small radar cross-sections, varying flight parameters, and close UAV trajectories make traditional methods difficult. Manual or fixed algorithm-based radar parameter and positional adjustments are limited. The proposed method uses RL to dynamically adjust radar parameters and position to improve swarm UAV detection, tracking, and classification. Simulations have shown that RL-based adjustments outperform other AI algorithms by learning the environment through rewards in real time. Khan et al. [27] examined how UAVs' high mobility, frequent topology changes, and 3D movement affected routing in flying ad-hoc networks (FANETs). Topology-based routing is becoming more important in FANET ad-hoc networking protocols. Throughput, end-to-end delay, and network load can be improved by topology-based routing protocols. That text briefly summarizes these protocols, highlighting their data exchange features and pros and cons. To evaluate topology-based routing protocols, the OPNET simulator was used.

Ortiz et al. [28] described the service quality of flying ad hoc networks' wireless drone swarm ad hoc networks. Drones can transmit and receive data like routers, functioning as hierarchical nodes for MPLS, FHAM, and IPv6 protocols. FANET quantifies many network performance metrics, including delay, jitter, throughput, lost packets, and sent/received packets. Various configurations of drones, including swarms consisting of 10, 20, 30, and 40 units, were subjected to testing. That study investigated the registration process and message sequences of swarm drones, analyzed end-to-end drone traffic, identified security vulnerabilities, and observed service patterns. Simulated measures, namely those indicating favorable outcomes, were employed to forecast trends in the actual world. The findings of this study provide insights into the anticipated behavior of the network in real-world scenarios, hence informing the forthcoming experimental investigation. Additionally, Ortiz et al. showcased the assessment of experience quality by utilizing service quality indicators obtained from mathematical models and effectively applied the quality-of-experience approach in the agricultural sector, mainly focusing on drones as the primary subject.

This article utilizes the MaLSTM algorithm to analyze previous flight data and trajectory predictions for accurate UAV path forecasting, based on the explanations provided regarding UAV-related research. Furthermore, the UAV's capacity to identify similarities between various flight paths can assist in preventing collisions, enabling it to promptly detect and mitigate potential dangers. Furthermore, MaLSTM can aid in mission planning by assessing the appropriateness of various flight paths in complex scenarios involving multiple UAVs. The role of anomaly detection in UAVs is to improve safety by identifying deviations from expected flight patterns. Integrating the MaLSTM algorithm into the UAV navigation system can enhance autonomous decision making by enabling UAVs to modify their routes based on the constantly evaluated resemblance to the expected pattern, thereby enhancing the performance of the network.

3. Proposed Method

3.1. Intelligent Packet Priority Architecture

In this work, when displaying communications from trusted nodes, we provide priority to the interface of trusted nodes. The intelligent packet priority module, also known as the IPPM, acts as a replacement for the default application priority that is used by the network. It is predicted that the interfering node will be effectively overwhelmed if priority is given to the trusted node. This allows the access point to successfully process the trusted packet ahead of the interfering packet, because the trustworthy packet was given priority.

The proposed algorithms, which are based on Ardupilot and NS3, were implemented in FlyNetSimulator, which was used to run the simulation. Figure 3 illustrates how we utilized the system architecture to operate the IPPM within the FlyNetSim framework. The NS-3 simulator was linked to the IPPM environment using the Python binding module (PBM). C++ was utilized during the development of the NS-3 interface, whereas Python was utilized during the development of the IPPM. Through the transfer of data facilitated by the PBM, the IPPM establishes a connection between the NS-3 and the MaLSTM models. Both parties have access to the PBM, which is primarily controlled in NS-3 and is accessible to them both. The network and topology configurations that produce training data for the MaLSTM model in IPPM were created with the help of the NS-3 simulator, which was used to configure the network and topology. The MaLSTM algorithm is utilized in order for the IPPM to provide the functions necessary for priority packet prediction.

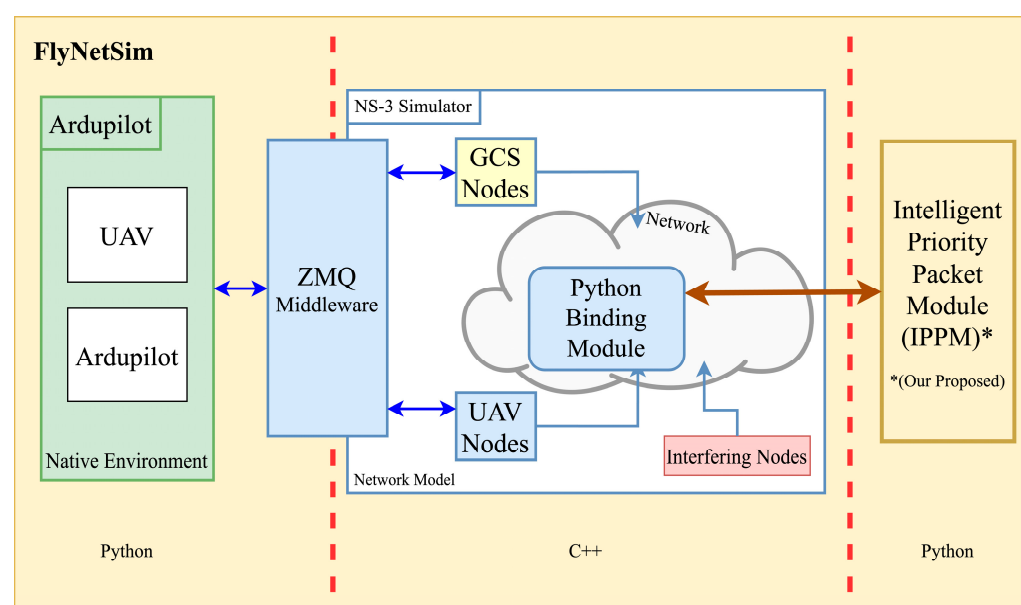


Figure 3. IPPM architecture.

3.2. IPPM Algorithm

This section is dedicated to discussing the IPPM algorithm. When transmitted, packets from the UAV, GCS, and interfering nodes travel through the base station. Figure 4 depicts IPPM's algorithm.

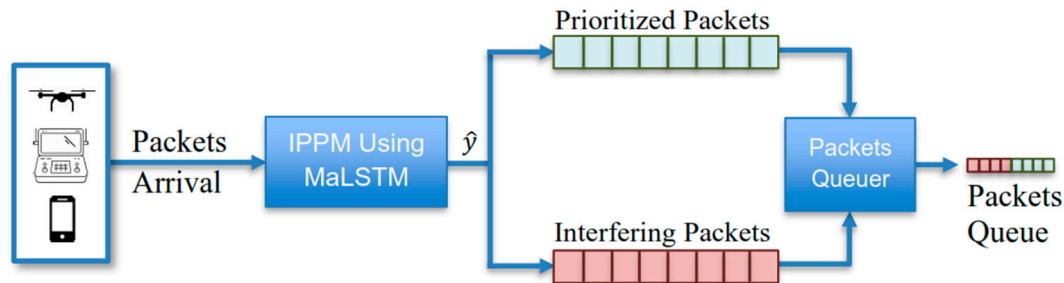


Figure 4. Proposed intelligent packet priority module (IPPM).

When UAV, GCS, and interfering nodes transmit packets, the base station's IPPM module classifies the packets. Priority packets can only be transmitted by UAV or the GCS, not by interfering nodes, which improves security by blocking unauthorized access from nodes outside the trusted UAV and GCS infrastructure, protecting critical communications from potential malicious attacks. The system optimizes performance and reliability by restricting priority packet transmission to UAVs and the GCS, which have specialized communication hardware and protocols for efficient handling of such traffic. Centralizing the transmission of priority packets to UAVs and the GCS enhances control and coordination of critical operations, facilitating efficient network resource management and traffic prioritization. Priority files are placed in the primary queue pool and sent first after classification. If there are remaining slots in the queue pool and no packet arrival updates have been received, the remaining slots are utilized by packets sent via the interfering node. The detail of the proposed algorithm is described in Algorithm 1.

Algorithm 1. Intelligent Packet Priority Module Algorithm

Input: Arriving packets

Output: Packets queue

- 1:
 - 2: Initialize PacketQueue
 - 3: Initialize PrioritizedPacketList
Initialize InterferingPacketList
 - 4:
 - 5: **Function** classify_packets(packet)
 - 6: **If** packet comes from UAV or GCS
 - 7: Predict priority using MaLSTM model
 - 8: **If** packet is prioritized
 - 9: Add packet to PrioritizedPacketList
 - 10: **Else**
 - 11: Add packet to InterferingPacketList
 - 12: **End If**
 - 13: **Else If** packet comes from an interfering node
 - 14: Add packet to InterferingPacketList
 - 15: **End If**
 - End Function
 - 16:
-

Algorithm 1. *Cont.*

```

17: Function process_packet_queue()
18: While PacketQueue is not empty
19:   packet <- get next packet from PacketQueue
20:   classify_packets(packet)
21: End While
   End Function

22:
23: Function send_packets()
24: For each prioritized_packet in PrioritizedPacketList
25:   send prioritized_packet
26: End For
   available_slots <- QueueCapacity - size(PrioritizedPacketList)

27: If available_slots > 0
28:   interfering_packets_to_send <- get first N packets from InterferingPacketList where N is
   available_slots

29:   For each interfering_packet in interfering_packets_to_send
   send interfering_packet
30:   End For
31: End If
32: End Function
33:
   While True
34:   process_packet_queue()
35:   send_packets()
36:   Wait for next packet arrival or time interval
37: End While
38:

```

3.3. Model Architecture

In recent years, there has been significant attention given to long-short-term memory (LSTM) [29], a powerful variant of a recurrent neural network (RNN) used in deep learning, due to its capability to model long-term dependencies and sequential data. In contrast to RNNs, LSTM is immune to the vanishing gradient issue [30]. Using its internal processes called gates, it controls the flow of information and keeps its state consistent throughout time. In particular, LSTM's track record of success in handling sequences of varying lengths was a major factor in our decision to use it. LSTM incorporates the following changes, given the input vector x_t , the hidden state h_t , and the memory state c_t :

$$h_t = o_t \odot \tanh(c_t) \quad (1)$$

$$f_t = \text{sigmoid}(W_f x_t + U_f h_{t-1} + b_t) \quad (2)$$

$$\hat{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (3)$$

$$c_t = i_t \odot \hat{c}_t + f_t \odot c_{t-1} \quad (4)$$

$$o_t = \text{sigmoid}(W_o x_t + U_o h_{t-1} + b_o) \quad (5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

where i_t , f_t , and o_t represent input, forget, and output gates, respectively, at time t . W_k , and U_k are LSTM parameterized weight matrices, b_k is the bias vector for each k in $\{I, f, c, o\}$, and \odot is the Hadamard product, which is merely an entrywise multiplication. The MaLSTM is a variant of the conventional LSTM in which the hidden state comparison between the two final layers is performed using the Manhattan distance formula as opposed to the cosine

or Euclidean distances. The Siamese design is characterized by the utilization of shared weights among subnetworks, resulting in a reduction in the number of parameters and mitigating the risk of overfitting.

Incorporating both the Siamese structure and the Manhattan distance, as its name suggests, the MaLSTM model is well named for its combination of these two features. The original objective of MaLSTM was to evaluate two sentences to determine whether they were semantically equivalent [31,32]. MaLSTM has been tailored to the context of packet identifiers in our proposed algorithm. The pairs of sentences become bundles. Siamese networks, like the one used by MaLSTM, have two identical parts ($LSTM_a$ and $LSTM_b$) that each take in a vector representation of two packets and produce a hidden state encoding a feature of the packets. As shown in Figure 5, a similarity measure was used to compare these hidden states and acquire a similarity score. Our proposed architecture operates by determining the similarity between incoming packets and $LSTM_a$ and comparing it with packets that should be prioritized by $LSTM_b$. A packet is classified as a prioritized packet if its similarity is 50% or higher. Conversely, packets with a similarity level below 50% are considered as packets sent from interfering nodes.

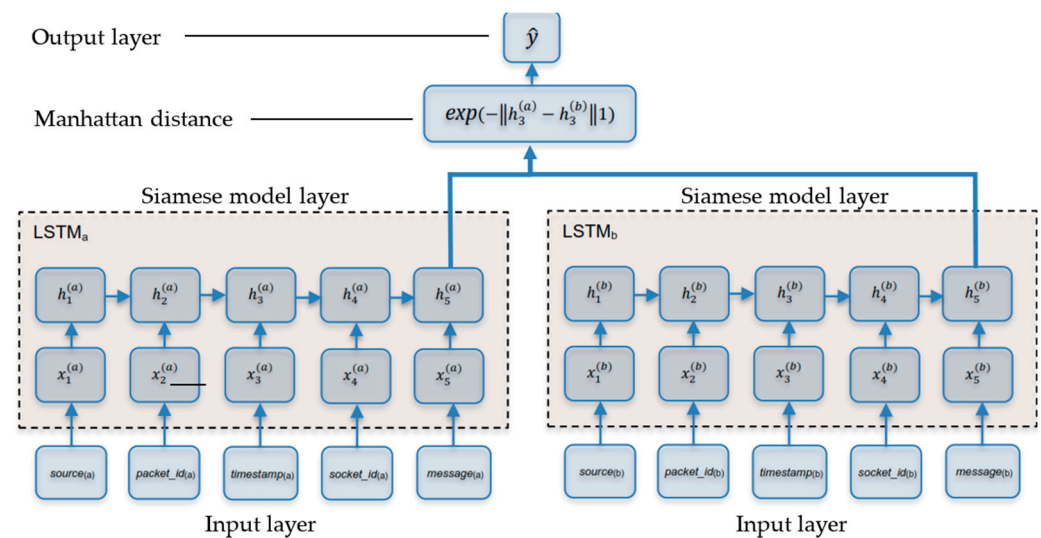


Figure 5. The MaLSTM architecture for the proposed IPPM.

3.4. Packet Pairs Dataset

To train the MaLSTM, we created a dataset of packet pairs that mimicked the structure of Zhang's [28] query retrieval dataset. We extracted the packets sent by connected nodes in the FlyNetSim network, i.e., UAV nodes, GCS nodes, and contending nodes, in order to construct the dataset. The collection contained 32,316 samples of packets. In Table 1, we describe the sample row of our dataset.

Table 1. Example Row of Packet Pairs Dataset.

Packet 1	@@@U_000***1***1671196580***472***0x5643cb247a40*** TELEMETRY:***last_heartbeat***0.149132145#7557#7570
Packet 2	@@@G_000***17***1671196612***80***0x5643cb247de0*** COMMAND: ***HEARTBEAT_MESSAGE
Label	1 (Similar)

Each sample was comprised of a pair of packets containing the ground truth regarding its similarity (1: similar, 0: dissimilar). Important packets from UAV and GCS nodes have distinct characteristics compared with packets from competing nodes. The characteristics

are described as packet source, packet identifier, packet timestamp, socket identifier, and command–telemetry message.

3.5. Model Training

Algorithm 2 describes the training process for the proposed model. We used the Adadelta method [32] for weight optimization to reduce the learning rate autonomously throughout model training. Gradient extraction with a threshold of 1.25 was also used to avoid the expanding gradient problem [33]. The size of our LSTM layers was 15, and the size of the embedding layer was 30. Mean squared error (MSE) was utilized as a standard regression loss function for prediction, and backpropagation was used to minimize cross-entropy loss with sample sizes as small as 64. To investigate the effect of threshold, we trained our model for 100 epochs with a wide range of values. The parameters were determined through empirical testing. Each parameter was adjusted independently on a development set in order to select the optimal value. Our model was created using Keras and PyTorch.

Mapping between the space of sequences of varying lengths d_{in} and a representation of hidden states (d_{rep}) of fixed dimension was achieved using LSTM. Each packet represented as a feature vector sequence (i.e., $Packet_1 = x_1, x_2, x_3, x_4, x_5$; $Packet_2 = x_1, x_2, x_3, x_4, x_5$) was supplied to the LSTMa and LSTMb, which updated their concealed state (h_a and h_b) at each sequence index point.

The final state of each LSTMa and LSTMb was a d_{rep} -dimensional vector, denoted by h_a and h_b in lines 7 and 8 of Algorithm 1. This vector contains the packet's characteristics:

$$\hat{y} = \exp\left(-\left\|h^{(LSTMa)} - h^{(LSTMb)}\right\|_1\right) \quad (7)$$

Similarity between pairs of sequences was calculated using the given network. Using the Manhattan similarity function, as indicated in line 9 of Equation (7), we determined the degree of similarity between the two vectors representing the underlying properties of each packet (h_a and h_b). The desired output of our model is to classify whether the predicted packet is similar to packets from important nodes (UAV and GCS) or not. Nevertheless, given the presence of a negative exponent, the Manhattan similarity function ($y_{\hat{}}$, or \hat{y}) is a continuous value between 0 and 1.

Algorithm 2. Model Training

Input: Packet pairs training set, epoch

Output: Trained model, loss, and accuracy history

- 1: **Packet1, Packet2** \leftarrow packet pairs from dataset
 - 2: **y** \leftarrow label of packet pairs
 - 3: **lstm** \leftarrow initiate LSTM(input_size, hidden_dim)
 - 4: **threshold** \leftarrow [0.1, 0.3, 0.5, 0.7, 0.9]
 - 5: **For each Epoch**
 - 6: Packet1, Packet2 \leftarrow embed(Packet1, Packet2)
 - 7: LSTMa, (h_a , ct) = self.lstm(Packet1)
 - 8: LSTMb, (h_b , ct) = self.lstm(Packet2)
 - 9: $y_{\hat{}}$ \leftarrow $\exp(-\text{abs}(h_a[-1] - h_b[-1]))$
 - 10: **For each threshold**
 - 11: optimizer.zero_grad()
 - 12: loss \leftarrow binary_cross_entropy($y_{\hat{}}$, y)
 - 13: loss.backward()
 - 14: optimizer.step()
 - 15: sum_loss \leftarrow sum_loss + (loss \times total_y)
-

Algorithm 2. *Cont.*

```

16: total  $\leftarrow$  total_y
17: is_simmilar  $\leftarrow$   $y_{\text{hat}} > \text{threshold}$ 
18: If is_simmilar = true
19:   correct  $\leftarrow$  correct + 1
20: EndIf
21: train_loss  $\leftarrow$  sum_loss/total
22: train_acc  $\leftarrow$  correct/total
23: EndIf
24: EndFor

```

To evaluate the training phase, it was necessary to apply a threshold value to separate between similar and dissimilar classes, as the value represented the degree of similarity of the packet. For example, as shown in line 17–20, if we set the class threshold of the MaLSTM as 0.5, then values of \hat{y} that are returned greater than 0.5 are considered a similar class. On the contrary, values of \hat{y} lower than or equal to 0.5 are considered a different class. The output of Algorithm 1 is a trained MaLSTM model, alongside the loss and accuracy history of each training epoch.

4. Results and Analysis

4.1. Simulation Setup

This section presents the simulation setup employed for the purpose of evaluation, as well as the evaluation and analysis of the MaLSTM model, including network delay, packet delivery ratio (PDR), and throughput. Our work was realized in the Linux Mint software (version 20.3) package FlyNetSim. Since NS-3 is a discrete event simulator that can simulate many networks, we found it to be the most suitable for testing the proposed IPPM in the noisy UAV-GCS setting.

We compared our method with our previous work [10] using a socket priority module and standard packet. The socket priority module works to reduce the delay caused by interfering nodes by first transmitting the payload with the highest priority. The IPPM algorithm provides better results for reducing delay and increasing PDR.

We built the simulation environment as shown in Figure 6. In this paper, we illustrate the whole communication scenario of our proposed IPPM, as depicted in Figure 6. The specific simulation parameters are shown in Table 2.

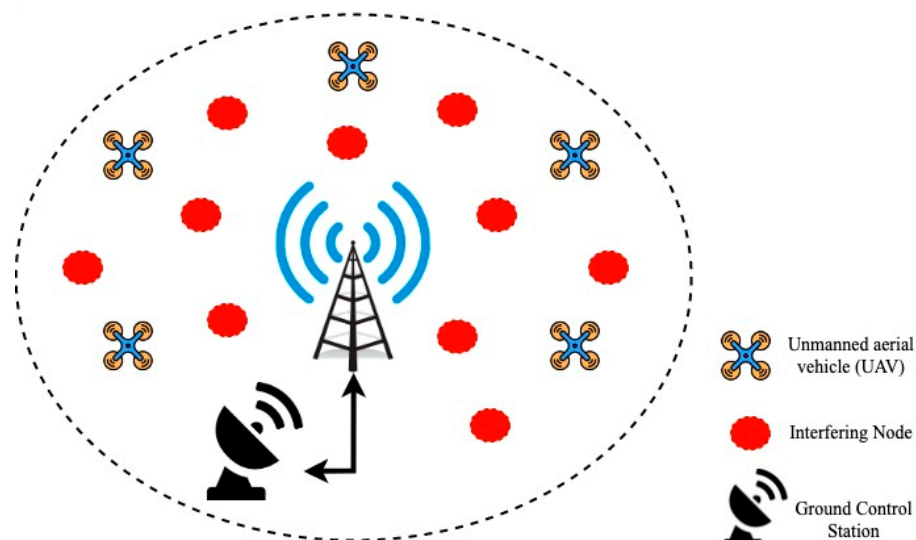


Figure 6. Simulation scenario.

Table 2. Simulation Parameters.

Parameters	Values
Network simulator	FlyNetSim (NS-3 and ArduPilot)
Network area (radius)	50 m
UAV transmission range (based on common LTE BTS)	100 m
UAV altitude	40 m
Number of UAV nodes	10 nodes
Number of interfering nodes (based on maximum devices allowed on common LTE BTS)	100
Number of channels	Single channel
Interfering nodes data rate	1 Mbps
UAV type	Swarm UAVs
Radio propagation channel model	Free space path loss (FSPL)

For simplicity, we considered 10 UAV nodes, 1 GCS node, and 100 interfering nodes in this simulation. We used FlyNetSim software (version 1.0) based on NS-3 and ArduPilot (version 4.5.3) to simulate the proposed approach and evaluate its performance. Table 2 shows a simulation parameter for the scenario with a UAV in a network area of 50 m radius. The free space path loss (FSPL) radio propagation channel model was considered in the simulation since it is useful and practical for evaluating communication range, link budget, and feasibility of swarm drone communications in obstruction-free environments. Users can efficiently assess the anticipated signal strength and coverage range for various UAV setups and operational circumstances. The IPPM algorithm can dynamically adjust packet prioritization by incorporating real-time feedback from the communication environment. If there is a decrease in the connection quality due to propagation loss or shadowing, the algorithm can prioritize packets to maintain the required quality of service.

Two traffic rate configurations were applied to test the module. We used an interfering traffic rate of 1 Mbps with a packet size of 100 bytes and 10 UAV nodes plus 100 interfering nodes. Simulation testing started with 10 interfering nodes and increased to 100 interfering nodes. The UAV altitude for the test was configured at 40 m with a transmission range of 50 m. We ran UAV and GCS simulations using FlyNetSim and then, the network part of the simulation was run on NS-3.

4.2. IPPM Evaluation

A set of 9.700 pairs (approximately 30% of total rows) were used for the evaluation phase. Our test sets were fed directly into MaLSTM. To evaluate the performance of our model, we compared the use of various similarity thresholds. We used accuracy as the evaluation metric.

Accuracy denotes the proportion of correctly classified packets as similar or dissimilar. Accuracy was measured in terms of correct and incorrect predictions for binary classification, as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

The model evaluation is described in Figure 7. Over 100 epochs, we found that the 0.5 threshold outperformed other threshold values. All similarity thresholds started the training with common accuracy values, ranging from 48% to 52% accuracy. However, the performance began to differentiate from epoch 30. The 0.5 threshold performance rose, peaking at 97.5% accuracy with the last training epoch, and was noted as the best performer. Meanwhile, the 0.3 threshold performance gradually decreased and reached only 17.7% accuracy. We noted that the middle upper threshold values (0.5, 0.7, 0.9) had relatively higher accuracy than the lower threshold value (0.1, 0.3). In this case, we also found that tightening or loosening the threshold resulted in lower model performance or accuracy. In lower threshold cases (0.1, 0.3), the performance might have been lower because the

false positives were higher. The packets that should have been detected as dissimilar were considered similar. In contrast, in higher threshold cases (0.7, 0.9), the performance might have been lower because the occurrence of false negatives was higher. The packets that should have been detected as similar were considered dissimilar. Hence, we conclude that the best threshold to separate between labels in this task was right in the middle of the range, not leaning towards a lower or higher boundary.

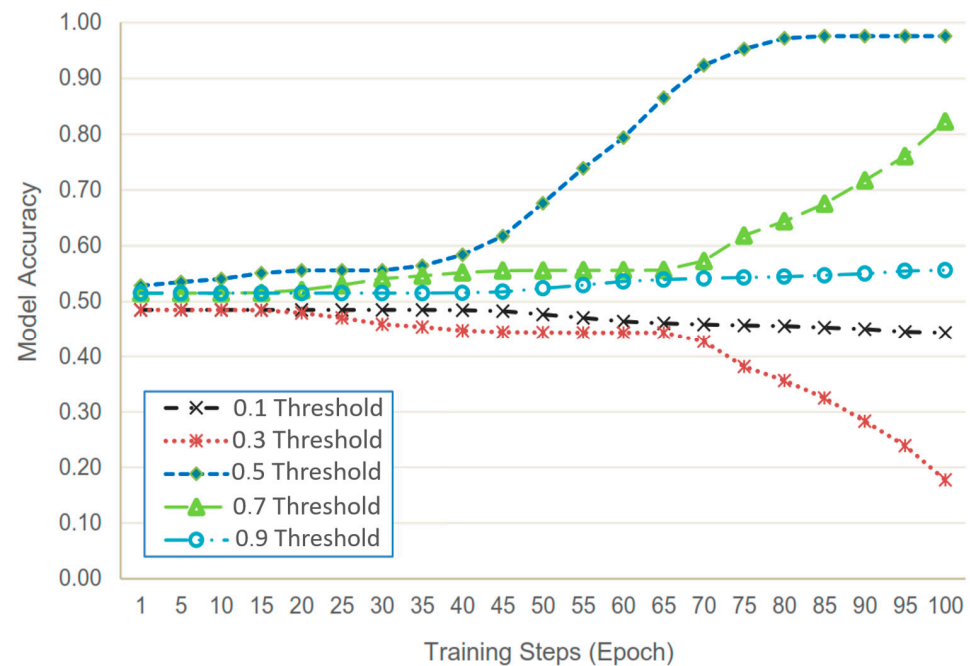


Figure 7. IPPM accuracy based on threshold.

4.3. Network Delay

The network delay evaluation results are described in Figure 8. We compare the network delay between our proposed IPPM with our previous work (socket priority module) and the standard packet model.

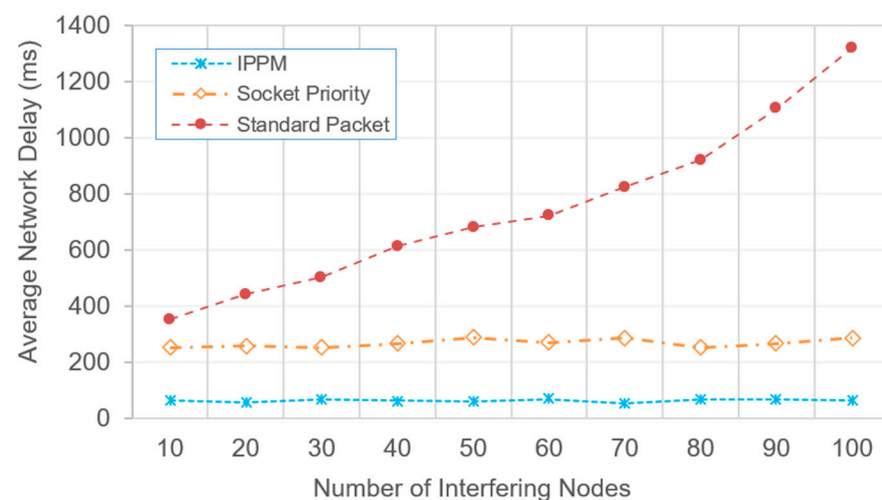


Figure 8. Network delay evaluation.

The results demonstrate that IPPM outperformed the socket priority module in reducing the average network latency during UAV-to-GCS packet transmission. IPPM had an average network delay of 62.2 ms, while the socket priority modules had an average

network delay of 267.1 ms. We noticed that the delay in the standard packet mechanism increased exponentially as the number of interfering nodes increased. The standard mechanism averaged 747.7 ms network delay. This might have happened because no module was applied to make important packets from the UAV and GCS be forwarded first.

4.4. Packet Delivery Ratio

Results of the study of the packet delivery ratio (PDR) are shown in Figure 9. Our suggested IPPM was compared to both our prior work and the standard packet method in terms of PDR. Equation (9) was used to determine PDR.

$$PDR = \frac{\text{Data Received}}{\text{Data Received} + \text{Data Loss}} \quad (9)$$

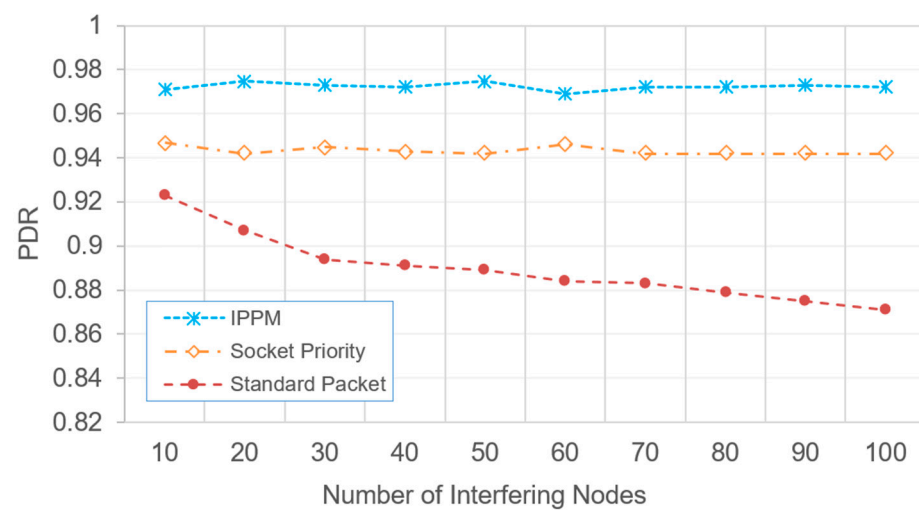


Figure 9. PDR evaluation.

According to these findings, IPPM worked better than either the socket priority module or the conventional mechanism in terms of PDR. The socket priority module and the regular packet method had an average PDR of 0.94 and 0.89, respectively, whereas IPPM had an average of 0.97. Our attention was drawn by the observation that the PDR for the typical packet technique rapidly diminished as the number of nodes contributing to interference increased. This may have occurred because there was no module applied to prioritize the forwarding of key packets from the UAV and GCS.

4.5. Throughput

Our suggested IPPM was compared to our prior work, and the developments show that IPPM has better performance according to the throughput testing. Figure 10 shows the results of the study of throughput. Equation (10) was used to determine throughput:

$$\text{Throughput} = \frac{\text{File Size}}{\text{Time}} \quad (10)$$

IPPM averaged 22.8 Gbps, while the socket priority module averaged 18.3 Gbps, respectively. Through analyzing the results of the performance tests, we observed that IPPM enhanced the overall performance quality of the UAV network as the number of interfering nodes increased.

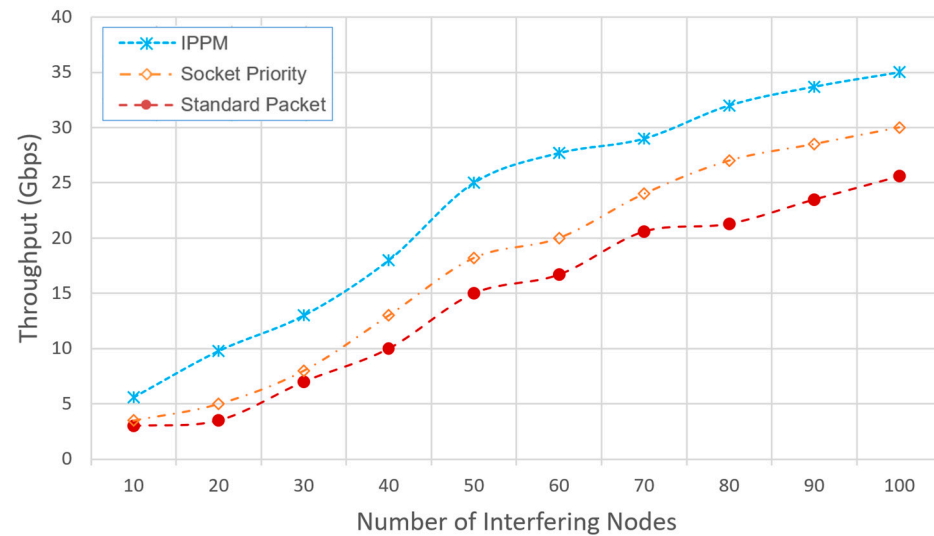


Figure 10. Throughput evaluation.

4.6. Algorithm Complexity Analysis

Two algorithms were analyzed in terms of their complexity. The first was that used in the model's training process, i.e., Algorithm 2. The training algorithm's complexity analysis involved considering key components such as input size, hidden dimension size, number of epochs, and thresholds. The algorithm begins by embedding the input packet pairs with an $O(n)$ time complexity. Subsequently, the packet pairs are passed through the LSTM, leading to a time complexity of $O(n * m * t)$, where n is the number of packet pairs, m is the hidden dimension size, and t is the number of time steps. The binary cross-entropy loss is then calculated with a $O(n)$ time complexity. Backpropagation involves updating the network's weights with a time complexity of $O(n * p)$, where p is the number of parameters in the network. Lastly, accuracy is determined by comparing predictions with thresholds, resulting in a time complexity of $O(n * t)$, where t is the number of thresholds. Overall, the algorithm's time complexity is $O(N * M * T)$, where N is the number of packet pairs, M is the hidden dimension size, and T is the number of thresholds, while its space complexity is $O(N * M)$. It is worth noting that this training process only occurs once to create the model.

The second analysis involved the proposed IPPM algorithm, i.e., Algorithm 1. The time complexity analysis of the proposed intelligent packet priority module (IPPM) pseudocode reveals that linear operations dominate the algorithm. The classification of individual packets is performed in constant time, $O(1)$, assuming the prediction time of the MaLSTM model does not increase with the number of packets. When processing the packet queue, the algorithm iterates through each packet once, leading to a linear complexity, $O(n)$, where n is the number of packets in the queue. Similarly, sending packets also has linear complexity, $O(n)$, since each packet is sent individually and the number of sending operations corresponds to the total number of packets, i.e., the sum of prioritized and interfering packets. Therefore, the main operational loop of the IPPM, which includes processing and sending packets, scales linearly with the number of packets, resulting in an overall time complexity of $O(n)$ for a single execution cycle. This analysis assumes efficient data structures for managing the packet queues, where insertion and removal operations are $O(1)$. If the data structures used for the packet lists have less optimal insertion and fewer removal complexities, this impacts the overall complexity of the algorithm. However, under the assumption of optimal data structures for queue management, the IPPM algorithm is efficient and scales well with increasing packets.

5. Conclusions

We developed an intelligent packet priority module (IPPM) in this paper. The IPPM can be used to classify the essential packets from UAV and GCS. For this purpose, we

introduced deep learning MaLSTM, which highly enhances the quality of network communications. Comprehensive simulations conducted using the FlyNetSimulator evaluated the performance of the suggested methods. The studies utilized the simulation of 10 UAVs and 100 interfering nodes across various scenarios. The experimental findings demonstrate that the suggested intelligent packet priority module (IPPM) improved network performance. The observed highly dependable packet delivery ratio (PDR), reduced latency, and enhanced network throughput support this method. In addition, we offer our viewpoint on the forthcoming developments in AI-based networking, with a primary focus on advancing techniques for vision-based tracking, enhancing environmental perception accuracy and utilizing lightweight deep learning algorithms suitable for onboard processing, AI-driven spectrum sharing, security-conscious networking, and vision-based tracking.

Author Contributions: D.B.P., J.H.W. and R.F.S. were involved in the conceptualization, methodology, investigation, and writing the original draft. D.B.P. and A.M. conducted the data curation, formal analysis, and visualization. R.F.S. was responsible for the conceptualization, resources, project administration, writing (reviewing and editing) and funding acquisition. R.N. was responsible for formal analysis, investigation, data curation, and writing (reviewing and editing). The initial draft of the article was authored by D.B.P., with subsequent versions being reviewed and commented on by all the contributors. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Seed Funding Professor Grant of the Faculty of Engineering, Universitas Indonesia, number: NKB-I958/UN2.F4.D/PPM.00.00/2022.

Data Availability Statement: The data and simulation code can be obtained by contacting the authors.

Conflicts of Interest: The authors do not have any pertinent financial or non-financial conflicts of interest to declare. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

IPPM	Intelligent packet priority module
UAV	Unmanned aerial vehicle
MaLSTM	Manhattan long short-term memory
GCS	Ground control system
TBS	Terrestrial base station
NS-3	Network Simulator-3
PBM	Python binding module

References

1. Bezmenov, V.; Safin, K.; Stepanov, S. Application of unmanned aerial vehicles for solving engineering tasks. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *890*, 12135. [\[CrossRef\]](#)
2. Erdelj, M.; Natalizio, E.; Chowdhury, K.R.; Akyildiz, I.F. Help from the Sky: Leveraging UAVs for Disaster Management. *IEEE Pervasive Comput.* **2017**, *16*, 24–32. [\[CrossRef\]](#)
3. Sona, G.; Passoni, D.; Pinto, L.; Pagliari, D.; Masseroni, D.; Ortuani, B.; Facchi, A. UAV multispectral survey to map soil and crop for precision farming applications. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.-ISPRS Arch.* **2016**, *XLI-B1*, 1023–1029. [\[CrossRef\]](#)
4. Shaikh, Z.; Baidya, S.; Levorato, M. Robust Multi-Path Communications for UAVs in the Urban IoT. In Proceedings of the 2018 IEEE International Conference on Sensing, Communication and Networking, SECON Workshops 2018, Hong Kong, China, 11–13 June 2018; pp. 1–5. [\[CrossRef\]](#)
5. Callegaro, D.; Baidya, S.; Ramachandran, G.S.; Krishnamachari, B.; Levorato, M. Information Autonomy: Self-Adaptive Information Management for Edge-Assisted Autonomous UAV Systems. In Proceedings of the IEEE Military Communications Conference MILCOM, Norfolk, VA, USA, 12–14 November 2019; pp. 40–45. [\[CrossRef\]](#)
6. Callegaro, D.; Baidya, S.; Levorato, M. Dynamic Distributed Computing for Infrastructure-Assisted Autonomous UAVs. In Proceedings of the IEEE International Conference on Communications, Dublin, Ireland, 7–11 June 2020; pp. 1–6. [\[CrossRef\]](#)
7. Henderson, T.R.; Lacage, M.; Riley, G.F.; Dowell, C.; Kopena, J. Network simulations with the ns-3 simulator. *SIGCOMM Demonstr.* **2008**, *14*, 527.

8. Ardupilot. Autopilot Suite Documentation. 2023. Available online: <http://ardupilot.com/> (accessed on 10 May 2023).
9. Baidya, S.; Shaikh, Z.; Levorato, M. FlynetSim: An open source synchronized UAV network simulator based on ns-3 and ardupilot. In Proceedings of the MSWiM 2018—Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Montreal, QC, Canada, 28 October–2 November 2018; pp. 37–45. [CrossRef]
10. Windiatmaja, J.H.; Tjahaja, J.C.; Al Amin, K.; Sari, R.F. Implementation of Socket Priority Module for Unmanned Aerial Vehicle Network using FlyNetSimulator. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1077*, 012021. [CrossRef]
11. H-SIM. H-SIM Flight Simulator. 2023. Available online: <http://www.h-sim.com> (accessed on 11 May 2023).
12. Erle Robotics. Erle-Brain | Erle Robotics. 2015. Available online: <http://erlerobotics.com/blog/erle-brain/> (accessed on 12 May 2023).
13. Meier, L.; Tanskanen, P.; Heng, L.; Lee, G.H.; Fraundorfer, F.; Pollefeys, M. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Auton. Robots* **2012**, *33*, 21–39. [CrossRef]
14. Navio. NAVIO2 Autopilot. 2023. Available online: <https://emlid.com/navio/> (accessed on 11 May 2023).
15. Ardupilot. Ardupilot SITL. 2023. Available online: <http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html> (accessed on 13 May 2023).
16. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. *ICRA Workshop Open Source Softw.* **2009**, *3*, 5.
17. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154.
18. Mozaffari, M.; Saad, W.; Bennis, M.; Nam, Y.-H.; Debbah, M. A tutorial on UAVs for wireless networks: Applications, challenges, and open problems. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 2334–2360. [CrossRef]
19. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.-C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3133–3174. [CrossRef]
20. Zhu, H.; Cao, Y.; Wang, W.; Jiang, T.; Jin, S. Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Netw.* **2018**, *32*, 50–57. [CrossRef]
21. Tang, F.; Kawamoto, Y.; Kato, N.; Liu, J. Future intelligent and secure vehicular network toward 6G: Machine-learning approaches. *Proc. IEEE* **2019**, *108*, 292–307.
22. She, C.; Sun, C.; Gu, Z.; Li, Y.; Yang, C.; Poor, H.V.; Vucetic, B. A tutorial on ultrareliable and low-latency communications in 6G: Integrating domain knowledge into deep learning. *Proc. IEEE* **2021**, *109*, 204–246.
23. Lei, L.; Tan, Y.; Zheng, K.; Liu, S.; Zhang, K.; Shen, X. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 1722–1760.
24. Jeganathan, A.; Dhayabaran, B.; Jayakody, D.N.K.; Ranchagodage Don, S.A. An Intelligent Age of Information Based Self-Energized Uav-Assisted Wireless Communication System. *IET Commun.* **2023**, *17*, 2141–2151.
25. Umair, M.B.; Iqbal, Z.; Bilal, M.; Almohamad, T.A.; Nebhen, J.; Mehmood, R.M. An efficient internet traffic classification system using deep learning for IoT. *arXiv* **2021**, arXiv:2107.12193.
26. Khawaja, W.; Yaqoob, Q.; Guvenc, I. RL-Based Detection, Tracking, and Classification of Malicious UAV Swarm through Airborne Cognitive Multibeam Multifunction Phased Array Radar. *Drones* **2023**, *7*, 470. [CrossRef]
27. Khan, M.A.; Khan, I.U.; Safi, A.; Quershi, I.M. Dynamic routing in flying ad-hoc networks using topology-based routing protocols. *Drones* **2018**, *2*, 27. [CrossRef]
28. Hamilton Ortiz, J.; Tavera Romero, C.A.; Taha Ahmed, B.; Khalaf, O.I. QoS in FANET Business and Swarm Data. *Comput. Mater. Contin.* **2022**, *72*, 1877–1899.
29. Zhang, W.-N.; Ming, Z.-Y.; Zhang, Y.; Liu, T.; Chua, T.-S. Capturing the semantics of key phrases using multiple languages for question retrieval. *IEEE Trans. Knowl. Data Eng.* **2015**, *28*, 888–900.
30. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.
31. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **1998**, *6*, 107–116. [CrossRef]
32. Mueller, J.; Thyagarajan, A. Siamese recurrent architectures for learning sentence similarity. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, VZ, USA, 12–17 February 2016; Volume 30.
33. Zeiler, M.D. Adadelata: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.