SPA: A Sense–Predict–Actuate TDMA Latency Reduction Scheme in Networked Quadrotors

Anandarup Mukherjee[1]*, Sudip Misra[1], and Narendra Singh Raghuwanshi[2]
[1]Department of Computer Science and Engineering, [2]Agriculture and Food Engineering Department
Indian Institute of Technology Kharagpur, India
Email: *anandarupmukherjee@ieee.org,

***Abstract*—In this paper, we propose the use of a Long Short-Term Memory (LSTM) based server-side sequence prediction algorithm to ease network data-load caused by rapid polling of multiple sensors onboard aerial robotic platforms, which are wirelessly tethered to a remote server for control and coordination. Our scheme reduces the network access time latencies between these platforms and the remote server hosting the control and scheduling mechanisms. Reduction in the TDMA-based access time is achieved by reducing the actual amount of data transmitted over the network, using partial transmission of actual sensor data over the network and server-side sequence prediction of the voluntarily missed sensor values. Our scheme allows the TDMA control of an increased number of networked platforms without change of infrastructure or the network characteristics.**

***Index Terms*—Quadrotor, Network congestion, Latency, LSTM, Flight parameters, Time Division Multiplexing.**

I. Introduction

In this work, a quadrotor UAV is wirelessly tethered to a remote access point connected to a server. This server and the network can support simultaneous connections from multiple such quadrotors. However, for proof of concept, we use a single one. The wirelessly tethered quadrotor's flight parameters such as roll (ψ), pitch (θ), yaw (ϕ), and thrust (T) are sampled onboard the quadrotor's computer and transmitted over the wireless channel to the server. The transmitted parameters, after appropriate processing, generates control signals and transmits them back to the quadrotor over the same channel. The tethering network has a fixed bandwidth due to constraints of the radios being used, which in addition to being used for transmitting/receiving quadrotor control and coordination signals, is used for transmitting multimedia data from the quadrotors. This reduces the number of simultaneous quadrotors that can be supported over the limited bandwidth. Approaches, such as Orthogoal Frequency Division Multiplexing (OFDM) [1] have shown promising results for simulations, however they tend to have higher data volumes, which has to be transmitted over the network, as well as handled by the remote server. Various delays associated with the overall implementation are segregated into three broad categories – quadrotor ($\delta_{quadrotor}$), network ($\delta_{network}$) and server ($\delta_{processing}$), as outlined in Fig. 1. However, some of these delays are further composed of smaller delays which can be given as $\delta_{quadrotor} = \delta_{processor} + \delta_{sensor}$. $\delta_{processor}$ and δ_{sensor} are attributed to the delays produced due to the quadrotor's onboard control hardware and data sampling from the quadrotor sensors. These two delays are integrated with $\delta_{quadrotor}$, which signifies the cumulative delay produced due to the limitations of the quadrotor's hardware. The delay parameter due to various channel effects of a mobile wireless sensor node (i.e., the quadrotor) is denoted by $\delta_{network}$. Similarly, the delay parameter $\delta_{processing}$ signifies the cumulative delays produced due to data-access at the remote server. $\delta_{processing}$ is represented as $\delta_{processing} = \delta_{DataAccess} + \delta_{LSTM}$, where $\delta_{DataAccess}$ and δ_{LSTM} are attributed to the delays produced as a result of data-access mechanisms at the server-end, and time taken to process and predict the data, respectively.

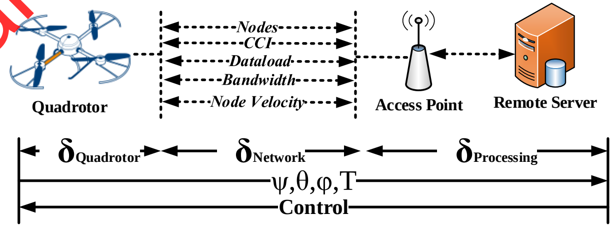


Fig. 1. Factors affecting a networked quadrotor system.

The quality-of-service (QoS) $\mathcal{Q}_{network}$, of the networked quadrotor, broadly depends on the bandwidth of the underlying network (B), the network data-rate (Δ_{node}), number of simultaneous connections to the network (k_c), velocity of nodes connected to the network (v_c), and co-channel interference (CCI), which is denoted by I_{cc}. According to Nyquist's theorem, B is related to the control and coordination signal data-rate Δ_{node} of a single network connected device (here, quadrotor) as $\Delta_{node} \leq 2B$. Additionally, Δ_{node} is represented in terms of quadrotor end-device sampling frequency (f_{node}), such that $f_{node} = \Delta_{node}^{-1}$. We assign the metrices D_L and C to represent the network data-load due to the transference of control data from the end-device to the server, and data-load from the end-device's applications (video and multimedia data), respectively, so that $D_L = k_c(\Delta_{node} + C)$. $\mathcal{Q}_{network}$ is be generalized as,

$$\mathcal{Q}_{network} \propto \frac{B}{(D_L + C)k_c v_c I_{cc}} \tag{1}$$

It is evident from Equation 1 for limited B, k_c is reduced as C increases. Our proposed scheme aims to alleviate this problem by increasing k_c for high values of C in a limited B network.

The velocity of the networked quadrotor (v_c), which behave as mobile nodes, of a wireless network also defines $\mathcal{Q}_{network}$, as it gives rise to certain unwanted effects such as multipath fading and Doppler shift. Considering the case when the quadrotor is airborne, away from obstacles and reflecting surfaces, multipath fading is ruled out. Hence, in our formulation, we only consider the effect of Doppler shift, as a result of the additional distance traveled by the EM wave during a mobile node's motion. For an angle, θ_d between the ground station and the airborne quadrotor flying with a velocity v_c (it is the same as the velocity of the node), having a wavelength of the EM wave as λ, the Doppler frequency f_d is denoted as $f_d = v_c \lambda^{-1} cos(\theta_d)$. As this work encompasses decreasing the TDMA access latency of networked quadrotors by decreasing the network data-load D_L caused due to transference of the quadrotors' flight parameters and control sequences, care is taken to ensure that the proposed solution does not affect other network parameters such as SNR, bandwidth, normal flight altitudes, trajectories, and especially, end-to-end network infrastructure.

Out of the five network delay parameters identified – $\delta_{processor}$, δ_{sensor}, $\delta_{network}$, $\delta_{DataAccess}$, δ_{LSTM} – the regions of improvement are restricted to reducing δ_{sensor} and δ_{LSTM}. The only network parameter defining the QoS, which is modifiable under the given conditions, is D_L. Reducing D_L translates to reducing f_{node}, so that the Δ_{node} generated from each quadrotor on the network is low, allowing for higher bandwidth utilization and incorporation of more quadrotors within the constrained bandwidth requirements.

Assumption 1. *The networked quadrotor is used for routine mission-based tasks with a fixed flight-path and trajectory, and the network under study does not suffer from serious interference and fading effects.*

The proposed approach insinuates dropping some of the transmitted values of the sampled quadrotor sensor values as illustrated in Fig. 2b, which are to be transmitted over the network, such that the reduction of information on the quadrotor side is compensated by LSTM predicted values on the server side. This enables us to acquire the quadrotor's original sampled signal on the server side, and at the same time, reduce the data load on the network. The recreated signal at the server-end, which consists of alternating actual sampled values is denoted by S_i, such that i starts at 1 and samples every other sensor output ($i+ = 2$). Similarly, the LSTM-based temporal predicted values are denoted by P_i. The recreated sequence of $S_1, P_1, S_2, P_2, \cdots$ is used for generating the quadrotor flight control sequences, which are then transmitted back to the quadrotor over the network.

This approach involves quantifying the parameters affecting normal flight-path of a quadrotor. Under ideal conditions,

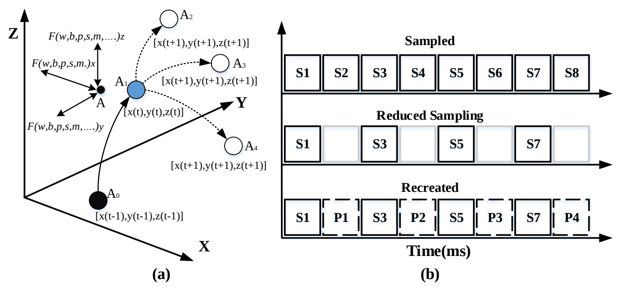


Fig. 2. (a) A model for the real-world path taken by the quadrotor and its consecutive temporal path estimation, (b) Time division multiplexed samples showing actual temporal signals inter-spaced with predicted signals using the SPA scheme.

and in the absence of external or internal intermittent disruptive factors, such as the effects of wind (w), quadrotor power levels (b), positional error due to sensor failures or GPS errors ($\mathcal{P}$), aerial obstacles (s), and other external factors (m) – the subsequent quadrotor flight step $(x(t+1), y(t+1), z(t+1))$ is determined using its current location coordinates at time t $(x(t), y(t), z(t))$ in conjunction with its velocity (v_q). The quadrotor flight-path is modeled as a Markovian process under ideal conditions, as in our approach, only the current state is required to predict the consecutive state of the quadrotor. In Fig. 2a, the spheres A_0 and A_1 represent a quadrotor's recent past ($T = t - 1$) and current ($T = t$) states, respectively. The possible future states ($T = t + 1$) of A_1 can be infinite. For the sake of representation, these states are denoted by A_2, A_3 and A_4. Under Markovian modeling, the change from A_0 to A_1 dictates the position of A_1 to be A_3, say. However, in real scenarios, the above-mentioned influencing factors cannot be neglected from quadrotor flight-path parameters. Due to the highly stochastic nature of some of the mentioned extraneous factors – $w, \mathcal{P}, s$ and m – a Markovian process based prediction of quadrotor's flight path fails. As represented in Fig. 2a, the stochastic nature of the extraneous factors implies that they can be along any axes and cause unpredictable changes in the future state of A_1. Further, it may cause A_1 to transition to any of the three states A_2, A_3, or A_4. In such scenarios, the use of deterministic approaches to predict the quadrotor's flight path, based on its previous flight patterns is appropriate. In the deterministic approach, we do not consider the effect of velocity, as it is a vector and the effect of extraneous factors (which are highly stochastic) may heavily influence the direction of predicted state from A_1, giving rise to erroneous results. The system is modeled as,

$$P(A_i|A_1)_{i=2,3,4} = P(x_k, y_k, z_k) \tag{2}$$

where, for transition from A_1 to the next state with n possibilities, the values of $P(x_k, y_k, z_k)$ are denoted as,

$$\begin{aligned} P(x_k, y_k, z_k) = max\{ \\ P(x_1(t), y_1(t), z_1(t)), \cdots P(x_n(t), y_n(t), z_n(t))\} \end{aligned} \tag{3}$$

At time t, the probability of a state in x, y or z axes being chosen over the other is determined by the prior choice of x, y or z at $t-1$, $t-2$, and so on. The LSTM-based sequence prediction operation in our scheme ensures this update for the task. This operation updates the prior probabilities at each time-step by means of choosing the most optimized weight vectors from the previous time-steps.

II. Related Works

The use of networked UAVs has cleared vast avenues for the use of regular UAVs beyond military and hobby flying. Vast amounts of work exist in the domain of networked UAVs, which deal with issues ranging from networked flight and formation control of UAVs [2] to enhancing communication coverage using UAVs [3]. Tran *et al.* [4] propose a resilient method of controlling networked multi-agent systems which promise better stability in their theoretical analysis, even under various external disturbances. Liao *et al.* [5] demonstrate a reconfigurable and distributed formation control scheme for UAVs, which ensures collision-free flight formation and flight formation reconfiguration. Li and Han [6] focus on packet delay minimization in multi-layer UAV networks by means of resource allocation optimization. Likewise, Koulali *et al.* focus on energy consumption optimization in UAV networks by employing a strategic game based activity scheduling scheme [7], and more recently, by choosing an optimal beaconing policy using Markov Decision Process [8]. Similarly, Guzey *et al.* [9] choose a hybrid consensus-based scheme for formation control of fixed wing UAVs. Other works on UAV networks encompass service oriented architectures for safe UAV flights, as proposed by Rodrigues *et al.* [10], UAV Software Defined Networks (SDN) and Network Function Virtualization (NFV), as demonstrated by White *et al.* [11].

Synthesis: Most of the targeted implementation zones for critical UAV applications (disaster-hit areas, surveying tasks in remote areas) and scenarios highlight the lack of proper network infrastructure for unhindered operation of these UAVs. To counter the network limitations and overload due to other factors such as – high mobility of the nodes – the existing approaches involve changes in the physical configuration of UAVs, its trajectory, its network or the overall network infrastructure. In contrast, our proposed solution of using deep learning-based sequence prediction would not only reduce the data-load on the supporting network but allow for more UAVs to be simultaneously associated with the same transmitting channel.

III. Flight Dynamics and Control

A quadrotor frame of reference is universally defined in terms of the North-East-Down (NED) coordinates, which are its inertial frame of reference. These NED coordinates, in conjunction with the body fixed coordinates (x, y, z), are used to determine the orientation vectors (ψ, ϕ, θ) of the quadrotor with respect to the inertial frame [12]. The roll, pitch, and yaw angles (ψ, ϕ, θ) determine the movement of the quadrotor about the x, y and z-axes, respectively. These x, y and z-axes, considered as the quadrotor fixed-body frame of reference, define the quadrotor's perfect linear position and denoted by β, such that, $\beta = \begin{bmatrix} x & y & z \end{bmatrix}^T$. The angular position – the attitude/heading of the quadrotor – is defined using Euler angles and is represented as η [12], which can be rewritten as $\eta = [\psi \quad \phi \quad \theta]^T$. For a vector $\hat{e}$ denoting the position of the center of mass of the quadrotor, we have, $\hat{e} = \begin{bmatrix} x & y & z \end{bmatrix}^T$. The non-linear dynamics of the quadrotor [12] is denoted by, $m\frac{d^2\hat{e}}{dt^2} = -mgD + RF$, where m is the mass of the quadrotor, g the acceleration due to gravity ($g \simeq 9.8m/s^2$), F the force acting on the quadrotor, and R the rotational matrix representing body frame of a quadrotor to the inertial frame of a quadrotor. For C_x denoting $cos(x)$ and S_x denoting $sin(x)$, R can be represented as,

$$R = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi S_\phi - S_\theta C_\phi & C_\theta S_\psi C_\phi + S_\theta S_\phi \\ S_\theta C_\psi & S_\theta S_\psi S_\phi + C_\theta C_\phi & S_\theta S_\psi C_\phi - C_\theta S_\phi \\ -S_\psi & C_\psi S_\phi & C_\psi C_\phi \end{bmatrix} \tag{4}$$

The rotation matrix R is orthogonal leading to $R^{-1} = R^T$ which represents the rotation matrix from the quadrotor's inertial frame to its body frame. Considering quadrotor motion along the z axis, F is considered to be composed of only one component such that, $F = \begin{bmatrix} 0 & 0 & T_f \end{bmatrix}^T$. T_f is the thrust or translation force, which relates to the gravitaional force of the motors (f_i) [12] as $T_f = \sum_{i=1}^{4} f_i$. For a motor constant m_c, the angular velocity of a specific rotor i is considered as ω_i and as it creates a force of f_i in the rotor axis direction, the torque, τ_i, during hover of the quadrotor, is expressed as, $\tau_i = m_c\omega_i^2 = m_c f_i$. The relation between moment of inertia (I_m) and total torque on the quadrotor (τ) for angular velocity in body frame of reference (Ω) is denoted by $I_m\frac{d\Omega}{dt} = -\Omega \times I_m\Omega + \tau$, where I_m is represented as,

$$I_m = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{5}$$

and, Ω is expressed as,

$$\Omega = \begin{bmatrix} \frac{d\phi}{dt} - \frac{d\psi}{dt}sin(\theta) \\ \frac{d\theta}{dt}cos(\phi) + \frac{d\psi}{dt}cos(\theta)sin(\phi) \\ \frac{d\psi}{dt}cos(\theta)cos(\phi) - \frac{d\theta}{dt}sin(\phi) \end{bmatrix} \tag{6}$$

Eventually, the generalized torques for the quadrotor, for distance between $\hat{e}$ and center of rotor represented as L, is represented as denoted in Equation 7 [12]. The dynamics of any given quadrotor UAV is nonlinear and at the same time coupled with each other and under-actuated, which makes control of this platform difficult.

$$\begin{bmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} -m_c & m_c & -m_c & m_c \\ -L & -L & L & L \\ -L & L & -L & L \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{7}$$

IV. Long Short-Term Memory

A Long Short-Term Memory (LSTM) consists of memory blocks, unlike traditional neural networks, which consists of only neurons. Each LSTM cell is a combination of specifically designed neural network units and operations, such as – *forget gate*, *input gate*, *output gate* and *candidate value update* – which take inputs from hidden layer h_{t-1} and C_{t-1} from previous LSTM units and an input x_t at an instant of time t to generate output h_t and C_t for the next upcoming LSTM unit. The figure shows the LSTM architecture used in our work. For weight W, a sigmoid activation function σ, a tanh activation function $tanh$, and a bias value b, the forget gate layer is expressed as, $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$. The forget gate layer outputs values between 0 and 1, which directs the LSTM unit with knowledge about the values to retain or drop from the previous state. Similarly, the input gate layer is responsible for selecting values to update, from the previous state, and is denoted as, $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$. The updated cell state C_t is given as, $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$, where $\tilde{C}_t$ is the vector of candidate values to be updated, which is expressed as $\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$. An output gate layer is also present, which is mathematically denoted as, $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$. This output gate layer function o_t, in conjunction with the updated cell state C_t, is used to determine the output h_t of the current (present) LSTM layer as $h_t = o_t * tanh(C_t)$, and is generalized according to our chosen LSTM architecture with 2 time-steps (elaborated in Section VI-A), which is represented as,

$$h_{t+1} = h_t + h_{t-1} + h_{t-2} \tag{8}$$

The consecutive sections on experimental setup and results elaborate upon the use of the LSTM-based quadrotor flight prediction model, and its integration with the controls of the quadrotor.

V. Experimental Setup

The quadrotor platform used is Crazyflie, which is a commercially available open-source platform. Besides being armed with various sensors, such as an accelerometer, a gyroscope, a magnetometer, and a barometer being sampled at a rate of 100 Hz, it is fully programmable. A 2.4 GHz radio interface with a channel bandwidth of 250 kbps, a packet size of 32 bytes, and encoded using Orthogonal Quadrature Phase Shift Keying (OQPSK), is used to establish a connection between the remote server and the quadrotor platform.

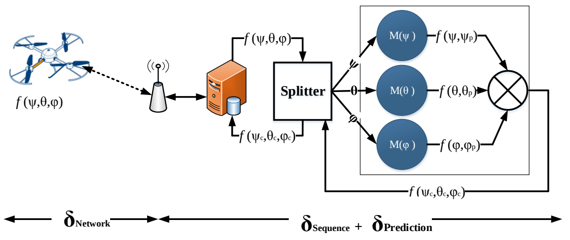
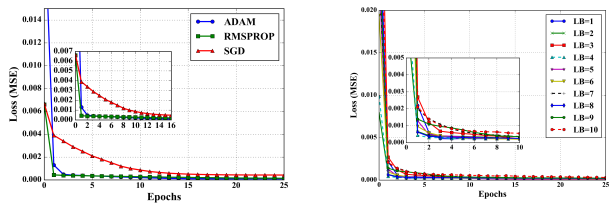


Fig. 3. The proposed system methodology, elaborating the server-side splitting and prediction model for the quadrotor flight-path prediction.



(a) Effect of optimizers (b) Effect of timesteps (t)

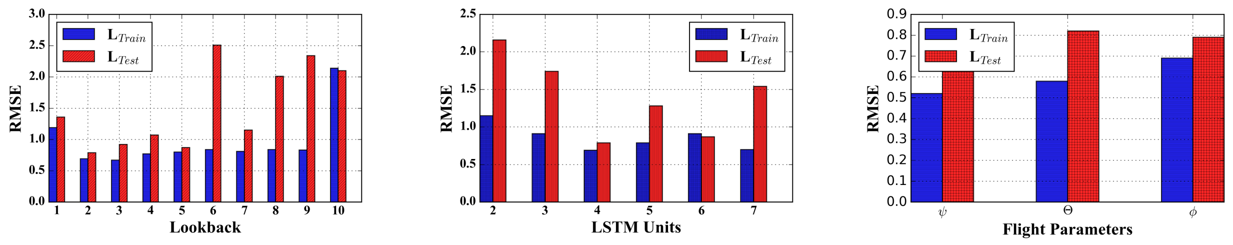
Fig. 4. Losses during training of LSTM models by means of variation of different parameters – optimizers, timesteps.

Sampled and packetized flight parameter data (ψ, θ, ϕ, T) from the quadrotor are transmitted over the network, eventually arriving at the remote server, as shown in Fig. 3. A data splitting module on this server discards the unwanted fields after extracting the flight parameters – ψ, θ, ϕ, T – from the received packet. As the thrust T of the quadrotor depends on the orientation vector of the quadrotor η, we designed the proposed scheme, *SPA*, to work only on η, and not T. The splitter module forwards the appropriate parameter to the correct LSTM model $(M(\psi), M(\theta), M(\phi))$ for future sequence prediction. The individual LSTM models output values such that, $M(\psi) \rightarrow f(\psi, \psi_p)$, $M(\theta) \rightarrow f(\theta, \theta_p)$, and $M(\phi) \rightarrow f(\phi, \phi_p)$. $f(\psi, \psi_p), f(\theta, \theta_p)$, and $f(\phi, \phi_p)$ are put in a control signal generation module, which generates appropriate control signals for the quadrotor, based on these new orientation vectors, η_c. Equation 8 is used separately for the three quadrotor flight parameters (ψ, θ, ϕ), such that,

$$\begin{aligned} h_{t+1}(\psi) &= h_t(\psi) + h_{t-1}(\psi) + h_{t-2}(\psi) \\ h_{t+1}(\theta) &= h_t(\theta) + h_{t-1}(\theta) + h_{t-2}(\theta) \\ h_{t+1}(\phi) &= h_t(\phi) + h_{t-1}(\phi) + h_{t-2}(\phi) \end{aligned} \tag{9}$$

where $h_{t+1}(\psi), h_{t+1}(\theta), h_{t+1}(\phi)$ are the predicted values at time-step $t+1$, henceforth denoted as ψ_p, θ_p, ϕ_p for the input values x_t, each corresponding to ψ, θ, ϕ at time-step t. η_c generates appropriate values of T, to maintain the mission or flight path of the quadrotor. It is noteworthy to mention that the output sequence generated from the LSTM models – $h_{t+1}(\psi)$, $h_{t+1}(\theta)$, $h_{t+1}(\phi)$ – are denoted by ψ, θ and ϕ respectively. The generated quadrotor controlling sequence is collectively denoted by $f(\psi_c, \theta_c, \phi_c)$ in Fig. 3, instead of T, for ease of understanding. $f(\psi_c, \theta_c, \phi_c)$ is re-packetized at the splitter and transmitted back to the quadrotor, which actuates the motors to act according to the received command, in turn modifying the quadrotor flight.

As we are focused on the remote server aspect of the overall architecture in this section, we denote the delay incurred by the data to travel from the quadrotor to the wireless access point by $\delta_{network}$, which is actually $\delta_{quadrotor} + \delta_{network}$. Additionally, the data delay from the access point to the control generation module is denoted by $\delta_{sequence} + \delta_{prediction}$.



(a) Effect of timesteps (t) (b) Effect of LSTM units (c) Effect on flight parameters

Fig. 5. Losses (RMSE) during training and testing of LSTM models by means of variation of different parameters.

$\delta_{sequence}$ signifies the delay due to data access and manipulation operations at the splitter module, whereas $\delta_{prediction}$ signifies the time taken to predict the future sequence of orientation vectors (η_c) and generate commands for the quadrotor, based on the incoming present values of η.

VI. Results and Discussion

In this Section, we describe the results obtained for the various stages of LSTM model generation, and the delay incurred by the proposed system in predicting flight parameters.

A. LSTM Parameter Selection

We adopted a standard train-test split ratio of $70-30$ for training our LSTM model. Fig. 4 shows the variation in the loss with respect to variations in parameters, such as time-steps (lookback), tested for a range between 1 to 60, and optimizers (ADAM, RMSprop, and SGD). The loss function chosen in this work is mean squared error (MSE), with a batch size of 1 and tested over 60 epochs. Fig. 4(a) shows that ADAM allows the architecture's training loss to rapidly converge to its local minima, as compared to RMSprop and SGD. This rapid convergence to minima establishes the superiority of ADAM over the other optimizers. Similarly, Fig. 4(b) shows the changes in MSE loss for variations in time-steps of the architecture (considering 4 LSTM units and ADAM as the selected optimizer). From this figure, it is concluded that the changes in loss are very close to each other for any conclusive indication of the superiority of one over another, and hence, the time taken by the architecture for providing output values becomes a deciding factor.

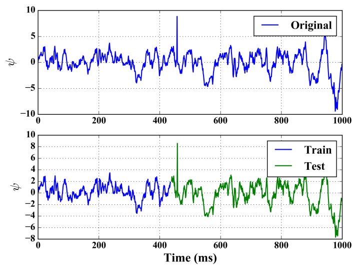
Fig. 5 shows the changes in root mean squared error (RMSE) for variations in time-steps from 1 to 10 (Fig. 5(a)) and LSTM units from 1 to 6 (Fig. 5(b)), for ADAM as the selected optimizer, during both training and testing stages of model generation for ψ over 20 epochs using MSE as the loss function. RMSE serves as an indicator of the usefulness of the trained model in the prediction tasks, as it checks the difference of the predicted signal from the expected signal – lower the difference, the better is the suitability of the model in the task prediction. In Fig. 5(a), a time-step value of 2 gives the least values of RMSE. Similarly, for Fig. 5(b), LSTM units of $3, 4$, and 6 give comparable results. Yet again, the prediction delay time ($\delta_{prediction}$) is chosen as the deciding factor for selecting an appropriate architecture amongst the three. Fig. 5(c) shows the change in training and testing loss for all three flight parameters. It is observed that the results of these three models are comparable.

B. System Latency

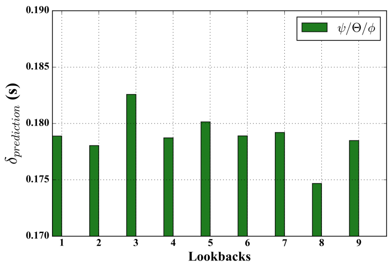
Fig. 6(a) shows the output sequences of the training and testing stages of the LSTM architecture. The top of this figure shows the original signal obtained from the quadrotor flight logs, whereas the bottom part represents the signal sequence output during training and testing stages of the LSTM model creation. It is to be noted that, as this scheme is dependent on historical values of flight parameters for training the model, its application, for now, is restricted to applications of quadrotors following a routine behavior. The outputs additionally show both the high and low-frequency changes in the quadrotor flight parameter signals are recreated very close to the original, signifying its suitability for use in our task. Fig. 6(b) shows the delay incurred by the model prediction unit (as shown in Fig. 3) for predicting signal sequence at $t+1$, against input signals from $t-t_0|_{t_0=0,1,2,3,\ldots}$. The value of t_0 depends on the chosen architecture and the time-step defined within it. The prediction delay bars shown in Fig. 6(b) are generated against a signal sequence consisting of 705 values, each of ψ, θ and ϕ. It is observed that the delays incurred for architectures having 1 and 2 LSTM units are comparable.

Fig. 6(b) shows the prediction delay incurred at the remote server for each of the three parameters – ψ, θ, ϕ – which is specific to the chosen LSTM architecture. The architecture with 4 LSTM units yields a slightly higher value of $\delta_{prediction}$ ($= 0.23s$ for 705 values) as compared to the other architectures. As 4 LSTM units provide the least RMSE in Fig. 5(b), this is the preferred choice for the final implementation of the SPA scheme.

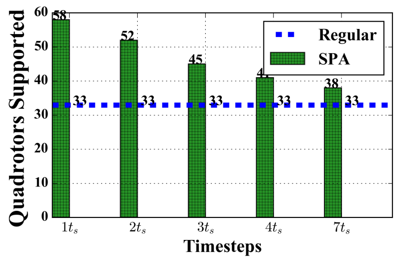
The $\delta_{prediction}$ values for all three parameters are collectively computed to 0.978 ms for each instance of incoming quadrotor data at the remote server. We approximate the $\delta_{prediction}+\delta_{sequence}$ value to 1 ms, which can be considered as the remote server's processing speed for the quadrotor data ($\simeq 1kHz$), which is attributed to the processing at the remote server. During regular operation, the quadrotor's



(a)



(b)



(c)

Fig. 6. Metrics highlighting the feasibility and advantages of the proposed method for use with networked quadrotors. (a) LSTM based temporal prediction of flight parameter – ψ – for both training and testing stages of the LSTM model generation. (b)Variations in the prediction delay (in seconds) – $\delta_{prediction}$ – for the quadrotor flight parameters (ψ, θ, ϕ) with respect to changing LSTM units. (c) Projected savings in network data-load represented in terms of number of additional quadrotors that can be controlled as compared to when the regular method is used.

sensor values are sampled at a rate of $30Hz$ (30 values per second), allowing for the control of $1000/30 \simeq 33$ quadrotors by utilizing the channel in a time-multiplexed manner. However, using the SPA scheme, the efficiency of the time-division multiplexing is substantially increased, as reported in Fig. 6(c). Fig. 6(c) shows only selected time-steps for an architecture with 4 LSTM units. The choice of time-steps in the final evaluation selection is made on the basis of Fig. 5(a), as these selected time-step/lookback values – $1, 2, 3, 4, 7$ reported the least RMSE amongst the lot. The reported improvements in the time-division multiplexing capabilities using the proposed SPA scheme is shown in Fig. 6(c).

VII. Conclusion

In this paper, we address the problem of decreased network throughput, and consequently, increased latency of the network due to excessive data-load transmitted between the ends. This gives rise to increased TDMA network latency. Further, this problem is accentuated in cases of networks, which are tasked with connecting actuating devices controlled remotely, the more complex the device is in terms of the number of sensors, the more is the data-load on the network. The proposed scheme of anticipating or predicting future value sequences, based on the values of incoming quadrotor UAV aerodynamic parameters (roll, pitch, and yaw) at the remote server is used for actuating the UAV. The parameter prediction is achieved using a deep learning architecture with LSTM units. The combination of actual and predicted values to estimate the flight path reduces the data load on the network, reduces the TDMA based network access latency of other quadrotors, and makes the network available to support even more quadrotor platforms, using the same previous specifications. This approach can be easily extended to other fixed path robotic platforms, which are controlled by a wireless network.

In the future, we plan to analyse the effects of this scheme on QoS and D_L, in addition to extending this scheme to incorporate multiple heterogeneous autonomous platforms with varying functionality, and with no pre-defined missions or trajectories.

Acknowledgment

This work is supported by Information Technology Research Academy (ITRA), Government of India under ITRA-Water Grant ITRA/15(69)/WATER/M2M/01

References

[1] V. Vahidi and E. Saberinia, "Orthogonal frequency division multiplexing and channel models for payload communications of unmanned aerial systems," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 1156–1161.

[2] E. Yanmaz, M. Quaritsch, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Communication and Coordination for Drone Networks," in *Ad Hoc Networks*, lecture notes of the institute for computer sciences, social informatics and telecommunications engineering ed. Springer, 2017, vol. 184, pp. 79–91.

[3] S. Rosati, K. KruÅijelecki, L. Traynard, and B. R. Mobile, "Speed-aware Routing for UAV Ad-hoc Networks," in *2013 IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, Dec 2013, pp. 1367–1373.

[4] D. M. Tran, T. Yucelen, and J. D. Peterson, "Resilient Control of Active-Passive Networked Multiagent Systems in the Presence of Persistent Disturbances," in *AIAA Guidance, Navigation, and Control Conference*, Texas, US, 2017, p. 1505.

[5] F. Liao, R. Teo, J. L. Wang, X. Dong, F. Lin, and K. Peng, "Distributed Formation and Reconfiguration Control of VTOL UAVs," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 1, pp. 270–277, Jan 2017.

[6] J. Li and Y. Han, "Optimal Resource Allocation for Packet Delay Minimization in Multi-layer UAV Networks," *IEEE Communications Letters*, vol. 21, no. 99, pp. 580 – 583, 2016.

[7] S. Koulali, E. Sabir, T. Taleb, and M. Azizi, "A Green Strategic Activity Scheduling for UAV Networks: A Sub-Modular Game Perspective," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 58–64, 2016.

[8] S. Koulali, M. Azizi, E. Sabir, and R. Koulali, "Optimal Beaconing Policy for Tactical Unmanned Aerial Vehicles," in *Advances in Ubiquitous Networking*. Springer, 2017, pp. 659–667.

[9] H. Guzey, "Hybrid Consensus-Based Formation Control of Fixed-Wing MUAVs," *Cybernetics and Systems*, vol. 48, pp. 1–13, 2017.

[10] D. Rodrigues, R. de Melo Pires, E. A. Marconato, C. Areias, J. C. Cunha, K. R. L. J. C. Branco, and M. Vieira, "Service-Oriented Architectures for a Flexible and Safe Use of Unmanned Aerial Vehicles," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 97–109, Spring 2017.

[11] K. J. White, D. P. Pezaros, E. Denney, and M. D. Knudson, "A Programmable Resilient High-Mobility SDN+ NFV Architecture for UAV Telemetry Monitoring," NASA, NASA Ames Research Center; Moffett Field, CA United States, Conference Pape 20170000332, Jan 2017.

[12] L. R. G. Carrillo, A. E. D. López, R. Lozano, and C. Pégard, "Modeling the Quad-rotor Mini-Rotorcraft," in *Quad Rotorcraft Control*. Springer, 2013, pp. 23–34.