**Modeling Control Delays for Edge-enabled UAVs in Cellular Networks**

<table>
<thead>
<tr>
<th>Journal:</th>
<th><em>IEEE Internet of Things Journal</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID</td>
<td>IoT-21075-2021</td>
</tr>
<tr>
<td>Manuscript Type</td>
<td>Regular Article</td>
</tr>
<tr>
<td>Date Submitted by the Author:</td>
<td>09-Nov-2021</td>
</tr>
<tr>
<td>Complete List of Authors:</td>
<td>Wu, Yu-Hsuan; National Yang Ming Chiao Tung University, Li, Chi-Yu; National Yang Ming Chiao Tung University, Lin, Yi-Bing; National Yang Ming Chiao Tung University, Wang, Kuochen; National Yang Ming Chiao Tung University, Wu, Meng-Shou; National Yang Ming Chiao Tung University</td>
</tr>
<tr>
<td>Keywords:</td>
<td>Vehicular Networks &lt; Sub-Area 2: Communications and Networking for IoT, Device-to-Device Communication &lt; Sub-Area 2: Communications and Networking for IoT, Efficient Communications and Networking &lt; Sub-Area 2: Communications and Networking for IoT</td>
</tr>
</tbody>
</table>
Modeling Control Delays for Edge-enabled UAVs in Cellular Networks

Yu-Hsuan Wu, Chi-Yu Li, Yi-Bing Lin Fellow, IEEE, Kuochen Wang, Meng-Shou Wu

Abstract—Real-time control solutions for unmanned aerial vehicles (UAVs) have attracted great interest in recent years. Most existing control methods use Wi-Fi technology. While Wi-Fi is inexpensive and easy-to-use, it has only a limited transmission range. Thus, 4G/5G cellular networks have been proposed as an alternative enabling technology. This study focuses on the problem of improving the appropriateness of the control commands sent by the ground control station (GCS) to the UAV over the control and non-payload communication (CNPC) link of the UAV through the cellular network. To satisfy the low-latency requirement of the CNPC link, multi-access edge computing (MEC) technology is leveraged to collocate the GCS and base station. The effectiveness of the proposed edge-based approach is demonstrated by conducting experiments on two LTE platforms with different MEC deployment methods. An edge-enabled UAV control solution is proposed in which each end-to-end control delay in the UAV-GCS system is estimated based on the preceding delay such that the location of the UAV at the moment it receives the control command from the GCS can be predicted in advance and taken into consideration by the GCS when formulating an appropriate control decision. To this end, an analytical modeling method is proposed for estimating the expected error range of each control delay based on a bimodal distribution approximation of the empirical control delays observed at the UAV. Finally, an event-driven simulator is developed to confirm the accuracy of the analytical predictions of the control delay based on the expected error between consecutive delays.

Index Terms—cellular network, UAV, MEC, edge computing

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have become increasingly common over the past decade for applications such as surveillance, search and rescue, aerial photography, and so on [1]. The total market opportunity for UAVs was estimated to be around 100 billion US dollars in 2020 [2], and is expected to increase in the coming years. UAVs have many favorable properties, including high maneuverability and agility, and are usually controlled remotely and in real-time using wireless communication technologies such as Wi-Fi and Zigbee. Among these technologies, Wi-Fi is generally the method of choice for most applications since it is available as a default on most smartphones and laptops [3]. However, while Wi-Fi is inexpensive and easy-to-use, it has only a short transmission range (e.g., few hundred meters). Consequently, 4G/5G cellular networks, with transmission ranges of up to 1 km or more, have attracted growing interest as an enabling technology for UAV wireless communications in recent years [4].

Many institutions, such as Qualcomm, the Federal Aviation Administration (FAA), and AT&T, have considered to optimize 4G LTE or 5G networks for UAV wireless communications [5]. In traditional cellular networks, the application servers are hosted in the cloud, and hence the UAV communications may suffer high communication delays. However, with the recent emergence of the multi-access edge computing (MEC) paradigm [6], cellular operators now have the option of collocating their MEC platforms with the base station (BS) and allowing service providers to install their application servers on them, thereby shortening the propagation delay and skipping network congestion in the Internet. As a result, MEC platforms appear to provide a highly promising approach for reducing the communication delay in UAV communications; particularly since MEC has been determined as a key component of emerging 5G networks [7].

As shown in Figure 1, the network architecture of UAV wireless communications includes two major communication links: one for control and non-payload communications (CNPC) and the other for data [8]. The CNPC link is used to exchange information between the UAV and the ground control station (GCS) to ensure safe, reliable and effective UAV flight operations. For example, the UAV sends sensed data or target tracking data to the GCS over the CNPC link, and the GCS then analyzes this data and responds to the UAV with an appropriate telecommand [9]. Meanwhile, the data link is used to support mission-related communications for the ground terminals. For example, when the UAV serves as a wireless BS or mobile surveillance camera, the data link carries the wireless backhaul traffic or video surveillance data. The latency requirement of the data link generally has a higher tolerance than the CNPC link since the timely control of the UAV through the CNPC link is one of the most critical aspects of the entire flight operation. For example,
in congested environments, delayed control commands may cause the UAV to collide with another vehicle operating in the same area.

This paper focuses on the CNPC link of a UAV-GCS network and aims to improve the quality of the control commands provided by the GCS to the UAV through an LTE network. To minimize the latency demand, it is assumed that the application server supporting the UAV control function (i.e., the GCS) is hosted by a MEC platform and is collocated with the BS. The latency performance is evaluated by conducting experiments on two practical LTE platforms with different MEC deployment methods and comparing the measured round trip times (RTTs) with those obtained from a traditional implementation with the GCS server located in the cloud. In the first deployment method, the MEC platform is implemented using a Bump-in-the-Wire technique on an open-source LTE core network with a commercial LTE small cell [10]. In the second method, the MEC is deployed using a Distributed S-GW (Serving Gateway) approach on the commercial Ericsson core network and BS. The empirical results confirm that edge-enabled UAV control indeed results in smaller RTTs than when the GCS server is located in the cloud.

Although the MEC-based CNPC links reduce the communication delay between the UAV and the GCS, non-negligible delays can still impact the quality of the GCS control commands. For example, UAVs usually have only limited computing resources, and thus when performing object identification or tracking tasks, they are usually guided by the GCS based on the sensed data which they provide at regular intervals. However, due to communication delays, the sensed data require a finite time to arrive at the GCS. Moreover, having processed the sensed data and formulated an appropriate response (e.g., a control command), the GCS incurs a further delay in communicating this command to the UAV. During both delay times, the UAV continues to move. In practice, a UAV traveling at a speed of 15 m/s may travel a distance of more than 1 m even after a short delay time of just 100 ms. Thus, if the UAV is to be properly controlled in performing its task, the GCS must formulate its control command decision based not on the position at which the UAV sensed the original data, but on that at which it will most likely be located at the moment it receives the command.

The present study refers to the RTT delay described above as the UAV control delay and defines this delay as the period between the moment at which the data are sensed by the UAV and the time at which the UAV receives the corresponding acknowledgment and control command from the GCS (see Figure 2). As discussed above, to properly navigate the UAV, the GCS needs to predict where the UAV will actually be located when it receives the control command. For most applications, this can be achieved simply by multiplying the average moving speed of the UAV by the control delay time. However, the effectiveness of such an approach is fundamentally dependent on the accuracy with which the delay time is known. When evaluating the control delays between the GCS and UAV, the present study ignores the packet processing time. Moreover, the video frame processing time is also ignored since, when encoding and decoding a 1080p video frame, the processing time can be less than 1 ms [11]. For simplicity, the control delay for each command is assumed to be equal to that of the previous (observed) command. However, there inevitably exists an error between the actual delay time of the control command and that of the previous command. Thus, to properly estimate the range of the current control delay, and formulate the control command accordingly, it is necessary to model this error in some way such that the observed control delay can be properly adjusted. In practical applications, the control delay is influenced by many factors, including the network traffic, signal interference, UAV movement, and so on. However, the changes in the control delay usually have trends and evolve gradually. Thus, it is both feasible and reasonable to predict the current control delay based on the previous observed delay and expected error range.

In this paper, the control delays collected from the two MEC platforms described above are approximated using bimodal distributions (i.e., mixtures of two Erlang distributions). It is shown that the bimodal distributions neatly fit the histograms of the empirical control delays. The approximated distributions are thus used to estimate the expected error between two consecutive control delays such that the delay time of the current control command can be updated accordingly. An event-driven simulator is developed to reproduce the empirical distribution of the control delays. The simulator is validated analytically and is then used to examine the relationship between the prediction accuracy of the analytical modeling approach and the expected error between consecutive control delays. The simulator is further used to examine the manner in which the expected error varies with the diversity of the two unimodal Erlang distributions within the bimodal distribution. It is found that as the diversity of the two unimodal distributions increases, the value of the expected error also increases. As a result, the prediction range of the control delay (and hence the predicted location range of the UAV) increases, and thus the GCS may act more conservatively in navigating the UAV in such a way as to avoid possible collisions and maintain the UAV on the proper trajectory.

The main contributions can be summarized as follows.

- First, the control delays between the UAV and the GCS are measured and compared for two practical MEC platforms given different signal strengths at the UAV.
The measured results not only serve as the basis for estimating the expected error between two consecutive control delays, but also provide a useful reference for practical edged-based UAVs.

- Second, a method is proposed for modeling the control delay between the UAV and the GCS by approximating a continuous probability distribution to the historical delay statistics. For each MEC platform, the approximated distribution is used to estimate the expected error between two consecutive control delays, and this error is then applied to the previous (observed) control delay at the UAV in order to predict the probable range of the upcoming delay. The predicted delay range can be used by the GCS to estimate the position of the UAV at the moment it receives the control command and to formulate the command decision accordingly.

- Third, an event-driven simulator is developed to model the bimodal control delay distributions of the two MEC platforms. Having validated the accuracy of the event-driven simulator analytically, it is used to confirm the effectiveness of the proposed control delay prediction method based on the expected error.

The remainder of this paper is organized as follows. Section II introduces the background and related work. Section III examines the communication delays of the UAV control messages in the two MEC platforms considered in the present study. Section IV proposes the modeling-based solution, and Section V introduces the event-driven simulator used to confirm its effectiveness. Section VI concludes this work.

II. BACKGROUND AND RELATED WORK

This section introduces the 4G LTE network architecture considered in the present study together with four MEC deployment options proposed by ETSI for LTE networks.

A. 4G LTE Network Architecture

Figure 3 presents a simple LTE network architecture consisting of an Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC). As shown, the E-UTRAN contains the User Equipment (UE) and Evolved Node B (eNB). The eNB provides the UE with data transmission, resource scheduling, and data buffering services during handovers, for example, and communicates with the UE through an LTE-Uu interface. The EPC contains the Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving Gateway (S-GW), and Packet Data Network Gateway (P-GW). In the control plane, the MME mainly takes care of the mobility management and performs user authentication with the HSS, which stores user profiles and connects with the eNB via an S1-MME interface. In the user plane, the S-GW interconnects the eNB and the P-GW via S1-U and S5 interfaces, respectively, and the P-GW forwards user data packets between the LTE network and the Internet.

The S1-U and S5 interfaces use the GPRS tunneling protocol (GTP) to perform user-plane data delivery [12]. Whenever a UE enables the mobile data service, an IP address is assigned and a default data bearer is created to carry data traffic. For each bearer, two GTP tunnels are set up for the S1-U and S5 interfaces, respectively, where each tunnel has a unique tunnel ID (TEID). When the eNB receives wireless data from the UE, it encapsulates the data into GTP packets and forwards them to the EPC via the S1-U interface. All the user-plane data sent to the eNB in the downlink direction are similarly encapsulated.

B. MEC Deployment Options

As shown in Figure 4, ETSI has proposed four possible deployment methods for MEC platforms at the edge of 4G LTE networks [10]: (1) Bump in the Wire; (2) Distributed S-GW with Local Breakout; (3) Distributed EPC; and (4) Distributed S/P-GW. The details of each method are briefly described in the following.

Bump in the Wire. The MEC is collocated with the eNB or between it and the EPC (i.e., sitting on the S1-U and S1-MME interfaces). The MEC routes plain IP packets to/from the application servers installed on it; while also forwarding GTP-encapsulated packets to/from the eNB and S-GW for control-plane and regular Internet traffic.

Distributed EPC. The MEC is co-located with the EPC, and the EPC is enabled to forward traffic to the MEC in addition to maintaining its original core components and functions.

Distributed S/P-GW. The EPC retains all its original components, but an additional set of S-GW and P-GW entities is deployed at the edge. The MEC routes, for example, the traffic that needs to be steered; in particular, the traffic can selectively reach either the EPC or the edge site.

C. Related Work

This study considers the problem of the MEC control of UAVs in cellular networks. Accordingly, this section of the paper briefly reviews previous work on both MEC-based and cellular-enabled UAVs.

MEC-based UAV research. Many studies on the application of the MEC concept to UAV networks have been performed in recent years. Broadly speaking, these studies can be classified into three main categories. First, the UAVs are enabled to
perform offload computing tasks (e.g., object recognition) to the ground MEC. Messous et al. [14] aimed to achieve a tradeoff between the task execution time and the UAV energy consumption by taking three computation options into consideration, namely on-board UAV resource, offloading to the MEC via the cellular network, and offloading to a Wi-Fi BS. Other studies examined the problem of cellular-connected UAVs continually offloading computation tasks to the BSs as they move along their trajectories. For example, Hua et al. [15] sought to minimize the UAV energy consumption by jointly optimizing portions of its offloading tasks and trajectory, while Cao et al. [16] aimed to minimize the mission completion time by jointly optimizing the UAV trajectory and computation offloading tasks.

Second, the UAVs are regarded as moving BSs that serve ground mobile devices in the MEC architecture. Zhou et al. [17] considered the UAVs to be capable not only of accepting computation offloading requests, but also of serving as energy transmitters for powering ground devices. The authors then sought to maximize the weighted sum computation bits by jointly optimizing the CPU frequency, offloading time, device transmission power, and the UAV trajectory. Jeong et al. [18] attempted to minimize the energy consumption of ground mobile devices by jointly optimizing the uplink/downlink communications and computations on the UAV under the constraints of service delays and the UAV energy budget.

Third, the UAVs are used to serve a MEC platform in providing computation offloading services to ground mobile/IoT devices with only limited computing resources [19]. The studies in [20], [21] sought to minimize the energy consumption of the ground mobile devices by optimizing the offloading task based on a joint consideration of the UAV position, computation task partition, and trajectory. Meanwhile, the studies in [22], [23] attempted to minimize the energy consumption of both the mobile devices and the UAV under the constraints of computation offloading, resource allocation, and flight trajectory. Zhang et al. [24] aimed to maximize the MEC system stability, while simultaneously minimizing its energy consumption and computation latency, by using a deep reinforcement learning method to optimize the UAV trajectory control and offload scheduling task. Zhang et al. [25] considered the mobile devices to be served not only by a flying UAV, but also by a ground BS, and attempted to minimize the latency and energy consumption of the system by offloading tasks between the UAV and the BS. Several studies [26]–[28] used multiple UAVs to serve as MEC/fog nodes for the provision of computing offloading services to ground mobile/IoT devices. For example, the study in [26] aimed to achieve load balancing while still maintaining the coverage constraint and service satisfaction, while that in [27] minimized the energy consumption of the ground devices by jointly optimizing the UAV task scheduling process, bit allocation, and trajectory. The study in [28] used multiple UAVs to construct a fog computing architecture and formulated the task allocation problem as that of minimizing the UAV energy consumption under the joint constraints of latency and reliability.

**Cellular-enabled UAV research.** Enabling cellular connectivity for UAV communications has attracted growing attention in industry and academia in recent years [4]. Many industrial institutions, including Qualcomm and AT&T, have expended significant effort in optimizing 4G/5G networks for UAV communications [5]. Lin et al. [29] considered the problem of enabling LTE connectivity for low-altitude small UAVs and presented several strategies for improving the LTE performance. Zeng et al. [8] surveyed the many challenges and opportunities of LTE-enabled UAVs. Khamidehi et al. [30] considered the trajectory optimization problem for multi-UAV-enabled multi-UAV-enabled cellular networks, in which the UAVs carried cellular BSs, and attempted to maximize the minimum data rate of the ground mobile users under the joint constraints of the UAV power, backhaul link capacity, and need for collision avoidance.

However, the literature lacks any study on control delay modeling over wireless dynamics for edge-based UAVs in cellular networks. This paper commences by considering the control delay behaviors of two practical MEC testbeds. The empirical delay data are then used for reference purposes in deriving an analytical model for estimating the expected error between consecutive control delays in order to improve the ability of the GCS to predict the location of the UAV and to formulate its control decision accordingly.

### III. Empirical Control Delays for Edge-enabled UAV

This section examines the practical communication delays of the UAV control messages on two MEC platforms, namely a Bump-in-the-Wire platform and a Distributed S-GW with Local Breakout platform. (Note that both platforms adopt LTE network architectures and are standard-compliant solutions without new functions deployed in the 3GPP components.) The discussions commence by describing the detailed architectures of the two platforms implemented in the present study. The corresponding experimental settings and empirical results are then briefly introduced and discussed.

#### A. MEC Platforms

The first MEC platform was a middlebox MEC developed by National Yang Ming Chiao Tung University (NYCU) in Taiwan [31]. The second platform was an intelligent MEC...
(iMEC), built by the Industrial Technology Research Institute (ITRI) of Taiwan [32], and also deployed at NYCU. Both platforms serve as MEC testbeds for commercial service trials and were selected in the present study for two main reasons: (1) they use two easily-deployed, standard-compliant methods, which can be widely adopted by other MEC platforms; and (2) the iMEC platform is deployed in the commercial Ericsson cellular infrastructure, and can thus be regarded as a reference MEC solution for the cellular industry.

For each platform, edge-enabled UAV control was enabled by deploying the UAV GCS as an application server on the platform. The details of the deployment process for the two platforms are described in the following.

1) Middlebox MEC Platform: The middlebox MEC adopts the Bump-in-the-Wire deployment method and hence serves as a middlebox sitting on the S1-U/S1-MME interface next to the eNB, as shown in Figure 5. Notably, it requires no modifications to the E-UTRAN or EPC, and only needs to handle the GTP tunnels over the S1 interface. This experimental testbed was deployed with an open-source LTE core, NextEPC [33], together with a commercial LTE small cell for the eNB.

In general, the middlebox platform provides MEC functionality via three major components. First, a GTP traffic engineering module encapsulates and decapsulates the GTP packets and steers the packets to either the MEC cloud or the E-UTRAN/EPC. The module additionally forwards MEC traffic to/from the application servers at the MEC cloud from/to the GTP tunnels, respectively, while also performing the usual message exchanges between the E-UTRAN and EPC in the control and data planes. Second, the platform uses the DNS service to redirect Internet application traffic to the MEC cloud. Specifically, it returns local IP addresses to DNS requests with the domain names which have their required traffic rules, and so on. iMEC consists of two main components in addition to the DNS service. First, it has an MEC management system that supports all automatic MEC management and configuration tasks. Second, it supports automatic deployment and removal of the servers, control and monitoring of their life cycles, installment of their required traffic rules, and so on. iMEC was leased from ITRI. For both platforms, an embedded engineering module encapsulates and decapsulates the GTP traffic to/from the GTP tunnels respectively, while also performing the usual message exchanges between the E-UTRAN and EPC in the control and data planes. Specifically, it returns local IP addresses to DNS requests with the domain names which have their required traffic rules, and so on. Fig. 6: iMEC platform based on the Distributed S-GW with Local Breakout method.

2) iMEC MEC Platform: The iMEC uses the Distributed S-GW with Local Breakout approach, which builds an S-GW next to the eNB in addition to the one in the EPC (see Figure 6). The local S-GW connects to the P-GW and eNB via S5 and S1-U interfaces, respectively. The platform requires the MME to perform S-GW selection such that the UEs are served by the local S-GW rather than the EPC S-GW. Therefore, the EPC needs to be aware of the MEC support with small operational updates. This platform was integrated with the Ericsson EPC and eNB platforms.

In general, the iMEC platform retains the original S-GW functions and builds an MEC platform on top of them. It provides a complete orchestration and deployment management service for the MEC application servers. Specifically, it supports automatic deployment and removal of the servers, control and monitoring of their life cycles, installment of their required traffic rules, and so on. iMEC consists of two main components in addition to the DNS service. First, it has an MEC management system that supports all automatic operations and provides users with a user interface to the MEC management and configuration tasks. Second, it supports two virtualization types for the MEC application servers: (1) container-based network function virtualization (NFV) with Kubernetes [34]; and (2) virtual machine (VM)-based NFV with Open Platform NFV (OPNFV) [35]. In the present study, the UAV GCS was deployed at a container on Kubernetes.

B. Experiment Settings

For the middlebox MEC platform, an LTE network was constructed consisting of a PC for the EPC, a commercial small cell for the eNB, and a server for the MEC. For the iMEC platform, the LTE network integrated with the iMEC was leased from ITRI. For both platforms, an embedded
TABLE I: Hardware equipment and software of middlebox and iMEC MEC platforms.

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Middlebox MEC</th>
<th>iMEC MEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPC</td>
<td>Intel Xeon E5-2680 (16GB RAM)</td>
<td>Intel Core i7-7700 (16GB RAM)</td>
</tr>
<tr>
<td>eNB</td>
<td>ERICSSON LTE EPC</td>
<td>NextEPC (Ubuntu 18.04 LTS)</td>
</tr>
<tr>
<td>MEC</td>
<td>Dell Poweredge R630</td>
<td>Huawei E3372h LTE dongle</td>
</tr>
<tr>
<td>UE</td>
<td>AAEON UP-Board Intel x5-z8350 (2GB RAM)</td>
<td>Ubuntu 16.04 LTS</td>
</tr>
<tr>
<td>UAV</td>
<td>DJI MATRICE 100</td>
<td></td>
</tr>
</tbody>
</table>

Intel x86 platform (AAEON UP-Board) was used as a UE to connect to the LTE network through an LTE dongle. As shown in Figure 7, the board was mounted directly on the UAV. Table I summarizes the hardware and software details of the two platforms.

In performing the experiments, the UE sent a 1 kB packet to the GCS, and the GCS replied with a control packet as soon as it received it. For each platform, the delay (i.e., RTT) was measured as the time between the moment at which the UE sent out the video frame and that at which it received the control packet. The RTT was measured for three different signal strengths at the UAV: strong, medium, and weak based on RSRP (Reference Signal Receiving Power) values of $>-90$ dBm, $[-105,-90]$ dBm, and $<-105$ dBm, respectively. Furthermore, for each MEC platform and every signal strength, measurements were obtained for two different GCS deployments, namely at the MEC and in the Internet cloud. For every experiment, 20000 RTT samples were collected, with an interval of 10 ms between them in every case.

C. Measured UAV Control Delays

Figure 8 shows the measured control delays of the UAVs for the middlebox MEC platform and cloud implementations, respectively, given the three different signal strengths. Figure 9 presents the corresponding results for the iMEC platform. For a strong signal at the UAV (Figures 8a and 9a), the RTTs of the UAVs are notably shorter for the MEC deployments of the GCS than for the cloud deployment. For example, in terms of the $50^{th}$ and $90^{th}$ percentiles, the middlebox MEC RTTs are 29.3 ms and 30.8 ms, respectively, and are around 23.3% and 33.2% quicker than those of the cloud RTTs (Figure 8a). Similarly, for the iMEC MEC, the corresponding RTT values are 15.5 ms and 22.1 ms, respectively, and are around 26.5% and 20.8% faster than those of the cloud. Comparing the two figures, it is found that the RTT values of the Ericsson eNB are around 38.3% lower than those of the WNC small cell.

For a medium signal strength (Figures 8b and 9b), the middlebox MEC RTT values are still lower than those of the cloud; however, the difference is less appreciable than in the case of a high signal strength. For the iMEC platform, the performance of the Ericsson eNB dominates the RTT value, and hence the RTT values are similar for both GCS deployment strategies. Finally, for a weak signal strength (Figures 8c and 9c), the MEC gains are offset by the poor wireless performance, and hence the RTT values of both MEC platforms are similar to those of the cloud.

IV. MODELING-BASED RANGE ESTIMATION ON UAV CONTROL DELAY

This section commences by describing the UAV control delay estimation problem considered in the present study. The modeling-based solution proposed for estimating the error
range of the control delay is then formulated. Finally, the application of the proposed modeling approach to the practical UAV operation is then briefly described.

A. Problem Statement

It is assumed that the UAV periodically reports its sensed data to the GCS through the CNPC link, and the GCS acknowledges the receipt of this data by sending back a packet, which may or may not contain control commands. The UAV measures the control delay between itself and the GCS for each round trip performed in this way, and then feeds this delay value back to the GCS in its next report. The GCS collects all the control delay data over time and uses this information to predict the upcoming delay. As shown in Figure 2, every time the GCS receives sensed data sent from the UAV at time \( t_0 \), it estimates a range of possible control delays \( t' = t_2 - t_0 \) at time \( t_1 \), and then uses this estimated range to predict the possible location range of the UAV at time \( t_2 \) as the product of the estimated control delay range and the UAV speed.

As described above, the GCS estimates the value range of each control delay based on a knowledge of the historical delay data. In practice, the control delay includes not only the RTT between the UAV and the GCS, but also the processing times of the sensed data and queuing times of the data packets at the two communication ends, as well as the packet transmission time. However, for simplicity, this section considers the empirical control delays obtained in Section III for illustration purposes. These delays refer to only minimal sensed data (e.g., only GPS and some inertial data). Hence, the processing and transmission times are both very small. Moreover, UAV traffic is regarded as high priority traffic by the cellular network, and consequently, the queuing time is also very small. As a result, the delay time is dominated by the RTT, and the other delay components are thus simply ignored. However, it is noted that the general modeling approach described in this section is expected to be equally applicable to other types of control delays, including those dominated by other factors, such as long processing times for image encoding and decoding.

B. Modeling-based Range Estimation

In the present study, the control delay is modeled by approximating a continuous probability distribution to the historical delay data over time. Suppose that the next and previous control delays are denoted as \( t' \) and \( t \), respectively, and the error between two consecutive delays is defined as \( \tau = |t' - t| \), where \( \tau \geq 0 \). The expected difference (i.e., error) between two consecutive delays, namely \( E[\tau] \), can then be derived from the approximated probability distribution and used to estimate the value range of the next control delay as \([t - E[\tau], t + E[\tau]]\).

The control delays are assumed to be i.i.d. random variables with a density function \( f() \). The density function can be derived by approximating the histogram of the measured delays. The expected value of \( \tau \) can then be formulated as:

\[
E[\tau] = \int_{t=0}^{\infty} \int_{t'=0}^{\infty} |t' - t| f(t') f(t') dt dt.
\]

For the purposes of the present study, the control delays are assumed to come from an Erlang distribution, where the shape and scale parameters of this distribution can be adjusted as required to approximate the histogram of the measured delays. Under stable network conditions, the delay distribution has a unimodal form; however, under more realistic conditions, an interplay inevitably exists between multiple delay factors, and hence the control delays are more reasonably modeled as a bimodal distribution (see Figures 8 and 9). Note that other distributions could also be considered for approximation purposes, provided that the approximated distributions fit the measured histogram well.

The expected errors between consecutive delays, \( E[\tau] \), for the unimodal and bimodal distribution approximations of the control delays can be derived as follows.

1) Control Delays with Unimodal Distribution: Without loss of generality, let \( t' = t + \tau \) and \( t = t' + \tau \) when \( t' \geq t \) and \( t' < t \), respectively. Furthermore, assume that \( t \) and \( t' \) are Erlang-distributed random variables such that

\[
f(t) = \frac{\lambda^k t^{k-1} e^{-\lambda t}}{(k-1)!},
\]

where \( k \in 1, 2, 3, \ldots \) and \( \lambda \geq 0 \).

Then, Equation 1 can then be rewritten as:

\[
E[\tau] = \int_{t=0}^{\infty} \int_{t'=0}^{\infty} \tau f(t) f(t + \tau) d\tau dt
+ \int_{t'=0}^{\infty} \int_{t'=t}^{\infty} \tau f(t') f(t' + \tau) d\tau dt',
\]

where the two terms on the right-hand side are equal since the two random variables, \( t \) and \( t' \), have the same distribution. As shown in Appendix A, Equation 3 can be rewritten as

\[
E[\tau] = 2 \sum_{j=0}^{k-1} \left( \frac{2k-j-2}{k-1} \right) \left( \frac{j+1}{2^k-j-1} \right).
\]

For the particular case of \( k = 1 \), \( t \) and \( t' \) are both exponential random variables. So, Equation 4 can be simplified as

\[
E[\tau] = \frac{1}{\lambda}.
\]

2) Control Delays with Bimodal Distribution: Observing the control delays collected from the middlebox MEC and MCE platforms in Section III, it is seen that the delay distributions for both platforms have a bimodal characteristic given a strong signal at the UAV (see Figures 8a and 9a). Moreover, the delay distribution for the middlebox MEC platform also has a bimodal characteristic given at medium strength signal at the UAV (see Figure 8b). As described in Section III, in the present study, the bimodal distribution of the control delays is modeled as a mixture of two Erlang distributions, \( f_1(t) \) and \( f_2(t) \), with random variables \( t_1 \) and \( t_2 \), respectively, i.e.,
The simulator was used to generate a series of control delays from the corresponding bimodal distribution approximations of the RTT measurement reported in Section III in accordance with the procedure shown in Appendix B.

For illustration purposes, consider the RTTs of the middlebox MEC platform with strong signals shown in Figure 8a. Dividing the bimodal distribution for the empirical RTTs into two unimodal distributions in accordance with the local minimum position yields $\pi = 0.23$. The mean and variance of the left unimodal distribution are then obtained as $\mu = 20.08$ and $\sigma^2 = 0.90$, respectively, while those of the right distribution are obtained as $\mu = 30.16$ and $\sigma^2 = 2.63$, respectively. As shown in Figure 10, the resulting approximated bimodal distribution of the control delays is in excellent qualitative agreement with the experimental histogram. Hence, the validity of the modeling approach is confirmed.

### C. Application of Modeling Approach to Practical Control Delay Estimation

Given a knowledge of the historical delay statistics over time, the modeling approach described above can be used to approximate the corresponding continuous probability distribution and to then derive the expected error between any two consecutive control delays, $E[\tau]$. During UAV flight operations, just before sending its sensed data to the edge server, the UAV estimates the range of the next control delay $t'$ as $[t-mE[\tau], t+mE[\tau]]$, where $t$ is the previous control delay and $m$ is a coefficient as a multiple of $E[\tau]$. It then sends this delay range, together with the sensed data, to the edge server. Given a knowledge of the average speed of the UAV, $\tau_m$, the GCS at the edge server uses the delay information to predict the probable movement distance of the UAV in the upcoming control delay $t'$ as $[\tau_m(t - mE[\tau]), \tau_m(t + mE[\tau])]$.

### V. Evaluation

An event-driven simulator was devised to observe the relationship between the prediction accuracy of the control delays (as evaluated using the modeling approach described above) and the expected error between consecutive delays, $E[\tau]$. The simulator was further used to examine the effects of the bimodal distribution parameters on the magnitude of $E[\tau]$. The analysis focused on the middlebox and iMEC platforms with a strong signal strength at the UAV. For both platforms, the simulator was used to generate a series of control delays from the corresponding bimodal distribution approximations of the RTT measurement reported in Section III in accordance with the procedure shown in Appendix B.

#### A. Validation of Event-driven Simulator

The validity of the event-driven simulator was evaluated using the analytical model derived in Section IV-B2 for the bimodal distribution of the control delay time. In particular, parameters $k_1$, $k_2$, $\lambda_1$, or $\lambda_2$ in Equation 6 were separately varied while fixing the other parameters. In each case, the error was calculated for all possible sets of two consecutive delays $\tau$ and the expected delay error, $E[\tau]$, was taken as the simulation result. Figure 11 compares the analytical and simulation results for the variation of $E[\tau]$ with $k_1$ and $k_2$, respectively. It is clear that the two sets of results are in excellent agreement with one another for both parameters. Overall, the average discrepancy between the analytical and simulation results over all the considered parameter settings was found to be less than 0.5%. Thus, the validity of the event-driven simulator was confirmed.

#### B. Prediction Accuracy

The bimodal distribution parameters, $\pi$, $k_1$, $\lambda_1$, $k_2$, and $\lambda_2$, were estimated for both MEC platforms (middlebox and
iMEC) under strong signal conditions. The event-based simulator was then used to generate a series of control delays for each platform. For each control delay $t$, the next delay $t'$ was evaluated analytically in the range of $[t - mE[\tau], t + mE[\tau]]$, where the expected error, $E[\tau]$, was derived from the approximated parameters and $m$ was assigned in the range of 0-3. The prediction accuracy of the analytical model was then evaluated by confirming whether or not the corresponding simulated delay value fell within the estimated range. The overall prediction accuracy of the model was finally obtained by averaging the prediction accuracy over all pairs of consecutive delays.

Figure 12a shows the variation of the prediction accuracy with the value of $m$ (i.e., a coefficient describing the multiples of $E[\tau]$ considered when estimating the next control delay). As expected, the prediction accuracy increases with increasing $m$ due to the corresponding increase in the control delay prediction range. For the middlebox MEC, the prediction accuracy increases from 95% to 99% when $m$ increases from 2.2 to 2.5. It is noted that when a larger $m$ results in a higher prediction accuracy for the control delay, the resulting broadening of the UAV location prediction range may cause the GCS to make more conservative decisions in order to prevent collisions.

C. Relationship between $E[\tau]$ and $\pi$

The effect of parameter $\pi$, i.e., the mixture coefficient of the bimodal distribution, on the expected error between consecutive delays, $E[\tau]$, was investigated by varying $\pi$ in the range of 0-1 in intervals of 0.1 while fixing all the other distribution parameters, and generating a series of control delays for each case using the validated simulator. Figure 12b presents the corresponding results for the two MEC platforms.

It is seen that $E[\tau]$ varies as a concave-downward function of $\pi$ in both cases. In other words, for $\pi = 0$ or 1, the bimodal distribution of the control delays reverts to a unimodal distribution. For the unimodal distribution, the control delay data are more densely concentrated, and hence the expected error of consecutive delays, $E[\tau]$, is smaller. By contrast, the largest value of $E[\tau]$ occurs at $\pi = 0.5$, for which the two unimodal distributions within the bimodal distribution have an equal share. For the middlebox MEC, $E[\tau]$ has values of 1.83, 1.08 and 5.84 for $\pi = 0, 1$ and 0.5, respectively.

Overall, the results shown that to minimize the value of $E[\tau]$, and thus narrow the prediction range of the UAV future location, the network provider should aim to stabilize the delays of the LTE network and MEC platform, respectively. In the ideal case, the control delays should be distributed with a unimodal distribution in order to achieve the smallest possible value of $E[\tau]$.

D. Relationship between $E[\tau]$ and Mean Distance between Mode Means

The effects of the diversity of the bimodal distribution on the expected error between consecutive delays, $E[\tau]$, was evaluated by varying the distance between the means of the two unimodal distributions in the bimodal distribution while retaining all the other distribution parameters unchanged. As shown in Figure 12c, $E[\tau]$ increases with an increasing separation of the means for both platforms. For the case where the two unimodal distributions have the same mean (i.e., the bimodal distribution has the form of a single unimodal distribution), the control delay data are most densely concentrated, and hence $E[\tau]$ has its minimal value. Conversely, as the distance between the two means increases, the control delays are more diversely distributed, and hence each control delay serves as a less reliable predictor of the following delay. Thus, the results indicate that, if a bimodal distribution of the control delays cannot be avoided, the network provider should aim to minimize the separation distance of the two means as much as possible in order to reduce the expected error between consecutive delays and hence improve the ability of the GCS to more accurately predict the expected location of the UAV at the moment it receives the upcoming control command.

VI. CONCLUSION

Cellular networks have the potential to equip UAV wireless control systems with not only a long-range communication capability, but also a low latency performance through the use of MEC technology. This study commenced by measuring the RTT between a UAV and an edge-based GCS on LTE platforms with two different MEC deployment methods (middlebox and iMEC). The results confirmed that the edge-based implementation of the GCS yielded an effective reduction in the mean RTT of the UAV-GCS system compared to that for a traditional cloud-based implementation. However, even though

Fig. 12: Relationship between the prediction accuracy, the expected delay difference $E[\tau]$, $\pi$, and the mean distance between two modes in the bimodal distribution.
the MEC-based deployment of the GCS reduces the control delay, obtaining precise estimates of each control delay is still essential in enabling the GCS to formulate appropriate control commands. Accordingly, a modeling approach has been proposed for approximating the historical control delays within the UAV-GCS system as a bimodal distribution. Given this distribution, the expected error range between consecutive control delays can be estimated by the UAV and then used by the GCS to predict the probable location range of the UAV. The validity of the proposed modeling approach has been confirmed using an event-based simulator. The results have shown that the ability of the GCS to predict the future position of the UAV can be improved by stabilizing the control delays within the UAV and MEC-enabled system. Overall, the results presented in this study provide a useful contribution to on-going research on edge-enabled UAV control.

**APPENDIX A**

**DERIVATION OF $E[\tau]$**

For the unimodal distribution described in Section IV-B1, the expected error between consecutive control delays (Equation 3) can be derived as follows:

$$E[\tau] = 2 \int_{t=0}^{\infty} \left[ \sum_{j=0}^{\infty} \frac{\lambda^j e^{-\lambda t} t^j}{j!} \right] d\tau$$

$$= 2 \int_{t=0}^{\infty} \left[ \sum_{j=0}^{\infty} \frac{\lambda^{j+1} e^{-\lambda t} t^{j+1}}{(j+1)!} \right] d\tau$$

$$= \sum_{j=0}^{\infty} \left[ \frac{\lambda^{j+1} e^{-\lambda t} t^{j+1}}{(j+1)!} \right]$$

$$= \sum_{j=0}^{\infty} \left[ \frac{\lambda^{j+1} e^{-\lambda t} t^{j+1}}{(j+1)!} \right]$$

$$= \sum_{j=0}^{\infty} \left[ \frac{\lambda^{j+1} e^{-\lambda t} t^{j+1}}{(j+1)!} \right]$$

$$= \sum_{j=0}^{\infty} \left[ \frac{\lambda^{j+1} e^{-\lambda t} t^{j+1}}{(j+1)!} \right]$$

**APPENDIX B**

**EVENT-DRIVEN SIMULATOR**

Figure 13 illustrates the simulation process used by the event-driven simulator to generate a series of control delays from a given bimodal distribution approximation of the empirical RTT results for a MEC platform. Based on the generated control delays, which are assumed to be observed over time, we can test the prediction accuracy. As shown, the simulator samples each control delay from one of two Erlang distributions (i.e., Erlang1 and Erlang2) based on their mixture coefficient, $\pi$. In each case, the number of collected delay values in the array `sim[]` (i.e., SAMPLE_SIZE), totals 100.

**REFERENCES**


Response to the Editor’s and Reviewers’ Comments

We thank the editor and the reviewers for their insightful comments, valuable suggestions, and great efforts and time. All these are very helpful for improving the clarity and quality of this new version. In this revised paper, we have made detailed explanations, and thorough revision in response to all the comments and suggestions. We have also improved the language and presentation throughout the draft from the help of an editor. For the sake of easy comparison, we provide two versions with and without highlighting our major revisions (in blue), which do not include the language/presentation improvement. In the following, we give our item-by-item response to each comment.

Associate Editor: Fernando, Xavier

Comments to Author:
My main concern is that the paper is difficult to read and understand because of poor English. Do a thorough proof reading and improve the language.

[Response] Thanks for this suggestion. We have improved the language and presentation throughout the help of an editor.

For instance, in many places it is not clear if the author wants to minimize the prediction error or to reduce the prediction range. For example, author says, 'the network provider should stabilize the delays of the LTE network to benefit the UAV with small prediction ranges of possible movement'. Why small prediction ranges? Elsewhere the author says, 'In this paper, we focus on estimating the value range of each control delay based on the history data (should be historical data)'. Anyway this makes more sense.

[Response] Sorry for the confusing writeup. We want to clarify that we propose an edge-enabled UAV control solution in which each end-to-end control delay in the UAV-GCS system is estimated based on the preceding delay such that the location of the UAV at the moment it receives the control command from the GCS can be predicted in advance and taken into consideration by the GCS when formulating an appropriate control decision. An analytical modeling method is proposed for estimating the expected error range of each control delay based on a bimodal distribution approximation of the empirical control delays observed at the UAV. We have the goal clearer throughout the paper.

Figure 11 caption: change to $E[\tau]$ variation with....

Correctness Validation --> Validation
[Response] Thanks for this suggestion. We have corrected them.

Reviewer: 1
Comments to the Author
This paper focuses on supporting the UAV’s control and non-payload communication (CNPC)
link. First, the authors introduced the multi-access edge computing (MEC) technology to collocate the UAV ground control station and the base station for minimizing latency. Furthermore, the authors proposed a modeling-based method to predict the error range of each control delay. However, some concerns are still on the way.

1. What role does the MEC play? Does the MEC act as a carrier only for operating the proposed algorithm? Pls clarify it further.

[Response] Sorry for the confusion. The MEC can be deployed by cellular operators and allow service providers to install their application servers on them. It serves as a network component to reduce communication delay for application servers. So, the application server for the UAV control with the proposed algorithm is one of the MEC-based application servers. We have clarified it in the second paragraph of the introduction section.

2. The authors aimed to predict the UAV’s location relying on the transmission delay. However, the transmission delay is greatly influenced by network traffic, inter-interference, and UAVs’ movement. Therefore, it seems non-practical for real scenarios.

[Response] Thanks for the comment. We want to clarify that it is required to predict the UAV’s location for its ground control station (GCS), since there is a delay between the image data sensed by the UAV and the control commands received by the UAV from the GCS. During this delay, the UAV is still flying, so the GCS should predict the probable range of the upcoming delay. The predicted delay range can be used by the GCS to estimate the position of the UAV at the moment it receives the control command and to formulate the command decision accordingly.

We agree that the transmission delay is greatly influenced by network traffic, inter-interference, and UAVs’ movement in real scenarios, but its changes have trends and gradually evolve. So, it is both feasible and reasonable to predict the current control delay based on the previous observed delay and expected error range. We have clarified it in the third from the last paragraph in the introduction section.

3. The transmission delay is related to the encoding & decoding scheme, radio propagation, and hardware configuration. Therefore, I doubt the feasibility that the UAV’s location can be predicted by using the transmission delay only.

[Response] Thanks for this comment. We want to clarify that during the transmission delay, the UAV continues to move. Given a knowledge of the average speed of the UAV, the GCS at the edge server can use the delay information to predict the probable movement distance of the UAV as the product of the estimated control delay range and the UAV speed.

4. The authors conduct a series of experiments to validate the effectiveness of the proposed scheme both on the iMEC and middlebox MEC. Is it scalable to other platforms? In fact, there are countless MEC platforms.

[Response] Thanks for this comment. We think that the proposed scheme can be applied to other MEC platforms, since the iMEC and middlebox MEC platforms are representative to them. The reasons are twofold. First, they use two easy-deployed methods, Bump in the wire and Distributed S-GW (Serving Gateway), from the four deployment options proposed by the ETSI standard. They could be widely adopted. Second, iMEC is deployed in the commercial Ericsson cellular infrastructure. It can be considered as a reference MEC solution for the cellular industry. We have clarified it in Section III.A.

5. This paper lacks discussion and analysis of the recent advances in this research area, especially
in the last three years. Pls improve it.

[Response] Thanks for this good suggestion. We have added more discussion about the research area involving UAV and MEC in the related work section, and included more than 10 references in the last three years.

6. As for MEC aided UAV, some novel recent works should be considered, such as ‘Multi-UAV-enabled load-balance mobile-edge computing for IoT networks’, ‘Taking Drones to the Next Level: Cooperative Distributed Unmanned-Aerial-Vehicular Networks for Small and Mini Drones’, ‘Distributed Fog Computing for Latency and Reliability Guaranteed Swarm of Drones’, etc.

[Response] Thanks for this suggestion. We have discussed these papers in the related work section.

Reviewer: 2

Comments to the Author

This paper studied the modeling of end-to-end transmission delays for edge-enabled UAV control in the cellular networks. Both the novelty and the presentation may not achieve to this journal. Comments and concerns are given below.

[Response] The novelty of this paper lies in the contribution that we propose a modeling-based method to predict the error range of each control delay for the UAV and also validate its effectiveness based on the traces collected from two practical MEC platforms. The prediction can make the UAV predict the future locations and take them into consideration in the control decision. Considering the future locations in the control decision is very critical, since different locations may need different control commands. Based on our literature survey, the control delay modeling over wireless dynamics for the edge-enabled UAV has not been addressed.

For the presentation, we have improved the language and presentation throughout the draft from the help of an editor.

The contribution of this paper is unclear. It is better to highlight what is the main contribution in this paper?

[Response] Sorry for the unclear writeup. We have listed three major contributions as follows in the introduction section.

- First, the control delays between the UAV and the GCS are measured and compared for two practical MEC platforms given different signal strengths at the UAV. The measured results not only serve as the basis for estimating the expected error between two consecutive control delays, but also provide a useful reference for practical edged-based UAVs.
- Second, a method is proposed for modeling the control delay between the UAV and the GCS by approximating a continuous probability distribution to the historical delay statistics. For each MEC platform, the approximated distribution is used to estimate the expected error between two consecutive control delays, and this error is then applied to the previous (observed) control delay at the UAV in order to predict the probable range of the upcoming delay. The predicted delay range can be used by the GCS to estimate the position of the UAV at the moment it receives the control command and to formulate the command decision accordingly.
- Third, an event-driven simulator is developed to model the bimodal control delay distributions of the two MEC platforms. Having validated the accuracy of the event-driven simulator analytically, it is used to confirm the effectiveness of the proposed control delay prediction method based on the expected error.
In addition, the latest references have been published in 2019. Usually, it is hard to see the novelty in this paper. In my opinion, related work should be discussed and some methods are needed to compare.

[Response] Thanks for this valid comment. We have discussed more related studies including more than 10 references in recent three years. We discover that the control delay modeling over wireless dynamics for the edge-based UAV has not been addressed in the literature.

What is the meaning of the APPENDIX B. It is necessary to explain in here.

[Response] We develop an event-driven simulator to generate a series of control delays from a given bimodal distribution, which is obtained from the approximation on the collected RTT results from the MEC platforms. Figure 13 illustrates the simulation process used by the event-driven simulator to generate a series of control delays from a given bimodal distribution approximation of the empirical RTT results for a MEC platform. Based on the generated control delays, which are assumed to be observed over time, we can test the prediction accuracy. We have added this explanation in Appendix B.

Algorithm description should be given.

[Response] Thanks for this good suggestion. The algorithm includes two parts: approximation of the bimodal delay distribution and prediction of the probable movement distance of the UAV. We have highlighted them in Sections IV-B3 and IV-C, respectively.

For the approximation of the bimodal delay distribution, Parameters $\pi$, $k_1$, $\lambda_1$, $k_2$, and $\lambda_2$ in Equation 6 are obtained via the following two-step procedure. First, the bimodal distribution is divided into two unimodal distributions at the point of the local minimum between the two peaks. The mixture coefficient, $\pi$, is then taken as the ratio of the left unimodal data portion to all the data. Second, the mean $\mu$ and variance $\sigma^2$ of each unimodal data portion are calculated and used to obtain $\lambda = \mu / \sigma^2$ and $k = \mu^2 / \sigma^2$, respectively.

For the prediction of the probable movement distance, given a knowledge of the historical delay statistics over time, the modeling approach described above can be used to approximate the corresponding continuous probability distribution and to then derive the expected error between any two consecutive control delays, $E[\tau]$. During UAV flight operations, just before sending its sensed data to the edge server, the UAV estimates the range of the next control delay $t'$ as $[t - mE[\tau], t + mE[\tau]]$, where $t$ is the previous control delay and $m$ is a coefficient as a multiple of $E[\tau]$. It then sends this delay range, together with the sensed data, to the edge server. Given a knowledge of the average speed of the UAV, $\bar{v}_n$, the GPS at the edge server uses the delay information to predict the probable movement distance of the UAV in the upcoming control delay $t_0$ as $[\bar{v}_n(t - mE[\tau]), \bar{v}_n(t + mE[\tau])]$. 
Modeling Control Delays for Edge-enabled UAVs in Cellular Networks

Yu-Hsuan Wu, Chi-Yu Li, Yi-Bing Lin Fellow, IEEE, Kuochen Wang, Meng-Shou Wu

Abstract—Real-time control solutions for unmanned aerial vehicles (UAVs) have attracted great interest in recent years. Most existing control methods use Wi-Fi technology. While Wi-Fi is inexpensive and easy-to-use, it has only a limited transmission range. Thus, 4G/5G cellular networks have been proposed as an alternative enabling technology. This study focuses on the problem of improving the appropriateness of the control commands sent by the ground control station (GCS) to the UAV over the control and non-payload communication (CNPC) link of the UAV through the cellular network. To satisfy the low-latency requirement of the CNPC link, multi-access edge computing (MEC) technology is leveraged to collocate the GCS and base station. The effectiveness of the proposed edge-based approach is demonstrated by conducting experiments on two LTE platforms with different MEC deployment methods. An edge-enabled UAV control solution is proposed in which each end-to-end control delay in the UAV-GCS system is estimated based on the preceding delay such that the location of the UAV at the moment it receives the control command from the GCS can be predicted in advance and taken into consideration by the GCS when formulating an appropriate control decision. To this end, an analytical modeling method is proposed for estimating the expected error range of each control delay based on a bimodal distribution approximation of the empirical control delays observed at the UAV. Finally, an event-driven simulator is developed to confirm the accuracy of the analytical predictions of the control delay based on the expected error between consecutive delays.

Index Terms—cellular network, UAV, MEC, edge computing

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have become increasingly common over the past decade for applications such as surveillance, search and rescue, aerial photography, and so on [1]. The total market opportunity for UAVs was estimated to be around 100 billion US dollars in 2020 [2], and is expected to increase in the coming years. UAVs have many favorable properties, including high maneuverability and agility, and are usually controlled remotely and in real-time using wireless communication technologies such as Wi-Fi and Zigbee. Among these technologies, Wi-Fi is generally the method of choice for most applications since it is available as a default on most smartphones and laptops [3]. However, while Wi-Fi is inexpensive and easy-to-use, it has only a short transmission range (e.g., few hundred meters). Consequently, 4G/5G cellular networks, with transmission ranges of up to 1 km or more, have attracted growing interest as an enabling technology for UAV wireless communications in recent years [4].

Many institutions, such as Qualcomm, the Federal Aviation Administration (FAA), and AT&T, have considered to optimize 4G LTE or 5G networks for UAV wireless communications [5]. In traditional cellular networks, the application servers are hosted in the cloud, and hence the UAV communications may suffer high communication delays. However, with the recent emergence of the multi-access edge computing (MEC) paradigm [6], cellular operators now have the option of collocating their MEC platforms with the base station (BS) and allowing service providers to install their application servers on them, thereby shortening the propagation delay and skipping network congestion in the Internet. As a result, MEC platforms appear to provide a highly promising approach for reducing the communication delay in UAV communications; particularly since MEC has been determined as a key component of emerging 5G networks [7].

As shown in Figure 1, the network architecture of UAV wireless communications includes two major communication links: one for control and non-payload communications (CNPC) and the other for data [8]. The CNPC link is used to exchange information between the UAV and the ground control station (GCS) to ensure safe, reliable and effective UAV flight operations. For example, the UAV sends sensed data or target tracking data to the GCS over the CNPC link, and the GCS then analyzes this data and responds to the UAV with an appropriate telecommand [9]. Meanwhile, the data link is used to support mission-related communications for the ground terminals. For example, when the UAV serves as a wireless BS or mobile surveillance camera, the data link carries the wireless backhaul traffic or video surveillance data. The latency requirement of the data link generally has a higher tolerance than the CNPC link since the timely control of the UAV through the CNPC link is one of the most critical aspects of the entire flight operation. For example,

Y.-H. Wu, C.-Y. Li, J. Y.-B. Lin, K. Wang, and M.-S. Wu are with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan (e-mail: jen3vn055@gmail.com, chiyuli@cs.nctu.edu.tw, liny@nctu.edu.tw, kwang@cs.nctu.edu.tw, and rio123148@gmail.com).
The present study refers to the RTT delay described above as the UAV control delay and defines this delay as the period between the moment at which the data are sensed by the UAV and the time at which the UAV receives the corresponding acknowledgment and control command from the GCS (see Figure 2). As discussed above, to properly navigate the UAV, the GCS needs to predict where the UAV will actually be located when it receives the control command. For most applications, this can be achieved simply by multiplying the average moving speed of the UAV by the control delay time. However, the effectiveness of such an approach is fundamentally dependent on the accuracy with which the delay time is known. When evaluating the control delays between the GCS and UAV, the present study ignores the packet processing time. Moreover, the video frame processing time is also ignored since, when encoding and decoding a 1080p video frame, the processing time can be less than 1 ms [11]. For simplicity, the control delay for each command is assumed to be equal to that of the previous (observed) command. However, there inevitably exists an error between the actual delay time of the control command and that of the previous command. Thus, to properly estimate the range of the current control delay, and formulate the control command accordingly, it is necessary to model this error in some way such that the observed control delay can be properly adjusted. In practical applications, the control delay is influenced by many factors, including the network traffic, signal interference, UAV movement, and so on. However, the changes in the control delay usually have trends and evolve gradually. Thus, it is both feasible and reasonable to predict the current control delay based on the previous observed delay and expected error range.

In this paper, the control delays collected from the two MEC platforms described above are approximated using bimodal distributions (i.e., mixtures of two Erlang distributions). It is shown that the bimodal distributions neatly fit the histograms of the empirical control delays. The approximated distributions are thus used to estimate the expected error between two consecutive control delays such that the delay time of the current control command can be updated accordingly. An event-driven simulator is developed to reproduce the empirical distribution of the control delays. The simulator is validated analytically and is then used to examine the relationship between the prediction accuracy of the analytical modeling approach and the expected error between consecutive control delays. The simulator is further used to examine the manner in which the expected error varies with the diversity of the two unimodal Erlang distributions within the bimodal distribution. It is found that as the diversity of the two unimodal distributions increases, the value of the expected error also increases. As a result, the prediction range of the control delay (and hence the predicted location range of the UAV) increases, and thus the GCS may act more conservatively in navigating the UAV in such a way as to avoid possible collisions and maintain the UAV on the proper trajectory.

The main contributions can be summarized as follows.
- First, the control delays between the UAV and the GCS are measured and compared for two practical MEC platforms given different signal strengths at the UAV.
The measured results not only serve as the basis for estimating the expected error between two consecutive control delays, but also provide a useful reference for practical edged-based UAVs.

- Second, a method is proposed for modeling the control delay between the UAV and the GCS by approximating a continuous probability distribution to the historical delay statistics. For each MEC platform, the approximated distribution is used to estimate the expected error between two consecutive control delays, and this error is then applied to the previous (observed) control delay at the UAV in order to predict the probable range of the upcoming delay. The predicted delay range can be used by the GCS to estimate the position of the UAV at the moment it receives the control command and to formulate the command decision accordingly.

- Third, an event-driven simulator is developed to model the bimodal control delay distributions of the two MEC platforms. Having validated the accuracy of the event-driven simulator analytically, it is used to confirm the effectiveness of the proposed control delay prediction method based on the expected error.

The remainder of this paper is organized as follows. Section II introduces the background and related work. Section III examines the communication delays of the UAV control messages in the two MEC platforms considered in the present study. Section IV proposes the modeling-based solution, and Section V introduces the event-driven simulator used to confirm its effectiveness. Section VI concludes this work.

II. BACKGROUND AND RELATED WORK

This section introduces the 4G LTE network architecture considered in the present study together with four MEC deployment options proposed by ETSI for LTE networks.

A. 4G LTE Network Architecture

Figure 3 presents a simple LTE network architecture consisting of an Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC). As shown, the E-UTRAN contains the User Equipment (UE) and Evolved Node B (eNB). The eNB provides the UE with data transmission, resource scheduling, and data buffering services during handovers, for example, and communicates with the UE through an LTE-Uu interface. The EPC contains the Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving Gateway (S-GW), and Packet Data Network Gateway (P-GW). In the control plane, the MME mainly takes care of the mobility management and performs user authentication with the HSS, which stores user profiles and connects with the eNB via an S1-MME interface. In the user plane, the S-GW interconnects the eNB and the P-GW via S1-U and S5 interfaces, respectively, and the P-GW forwards user data packets between the LTE network and the Internet.

The S1-U and S5 interfaces use the GPRS tunneling protocol (GTP) to perform user-plane data delivery [12]. Whenever a UE enables the mobile data service, an IP address is assigned and a default data bearer is created to carry data traffic. For each bearer, two GTP tunnels are set up for the S1-U and S5 interfaces, respectively, where each tunnel has a unique tunnel ID (TEID). When the eNB receives wireless data from the UE, it encapsulates the data into GTP packets and forwards them to the EPC via the S1-U interface. All the user-plane data sent to the eNB in the downlink direction are similarly encapsulated.

B. MEC Deployment Options

As shown in Figure 4, ETSI has proposed four possible deployment methods for MEC platforms at the edge of 4G LTE networks [10]: (1) Bump in the Wire; (2) Distributed S-GW with Local Breakout; (3) Distributed EPC; and (4) Distributed S/P-GW. The details of each method are briefly described in the following.

**Bump in the Wire.** The MEC is collocated with the eNB or between it and the EPC (i.e., sitting on the S1-U and S1-MME interfaces). The MEC routes plain IP packets to/from the application servers installed on it; while also forwarding GTP-encapsulated packets to/from the eNB and S-GW for control-plane and regular Internet traffic.

**Distributed EPC.** The MEC is co-located with the EPC, and the EPC is enabled to forward traffic to the MEC in addition to maintaining its original core components and functions.

**Distributed S/P-GW.** The EPC retains all its original components, but an additional set of S-GW and P-GW entities is deployed at the edge. The MEC is deployed next to the additional P-GW. However, to enable the eNB to connect to the local S-GW, the MME function is required to perform local S-GW selection [13].

**Distributed S-GW with Local Breakout.** Only the S-GW is co-located with the MEC platform at the edge. This option is similar to the third option in terms of the need for local S-GW selection, but provides a greater control on the granularity of the traffic that needs to be steered; in particular, the traffic can selectively reach either the EPC or the edge site.

C. Related Work

This study considers the problem of the MEC control of UAVs in cellular networks. Accordingly, this section of the paper briefly reviews previous work on both MEC-based and cellular-enabled UAVs.

**MEC-based UAV research.** Many studies on the application of the MEC concept to UAV networks have been performed in recent years. Broadly speaking, these studies can be classified into three main categories. First, the UAVs are enabled to...
considered the mobile devices to be served not only by a flying trajectory control and offload scheduling task. Zhang et al. [25] used a deep reinforcement learning method to optimize the UAV's energy consumption and computation latency, by using the MEC system stability, while simultaneously minimizing and flight trajectory. Zhang et al. [24] aimed to maximize the mission completion of both the mobile devices and the UAV under the constraints of service delays and the UAV energy budget.

Second, the UAVs are regarded as moving BSs that serve ground mobile devices in the MEC architecture. Zhou et al. [17] considered the UAVs to be capable not only of accepting computation offloading requests, but also of serving as energy transmitters for powering ground devices. The authors then sought to maximize the weighted sum computation bits by jointly optimizing the CPU frequency, offloading time, device transmission power, and the UAV trajectory. Jeong et al. [18] attempted to minimize the energy consumption of ground mobile devices by jointly optimizing the uplink/downlink communications and computations on the UAV under the constraints of service delays and the UAV energy budget.

Third, the UAVs are used to serve a MEC platform in providing computation offloading services to ground mobile/IoT devices with only limited computing resources [19]. The studies in [20], [21] sought to minimize the energy consumption of the ground mobile devices by optimizing the offloading task based on a joint consideration of the UAV position, computation task partition, and trajectory. Meanwhile, the studies in [22], [23] attempted to minimize the energy consumption of both the mobile devices and the UAV under the constraints of computation offloading, resource allocation, and flight trajectory. Zhang et al. [24] aimed to maximize the MEC system stability, while simultaneously minimizing its energy consumption and computation latency, by using a deep reinforcement learning method to optimize the UAV trajectory control and offload scheduling task. Zhang et al. [25] considered the mobile devices to be served not only by a flying UAV, but also by a ground BS, and attempted to minimize the latency and energy consumption of the system by offloading tasks between the UAV and the BS. Several studies [26]–[28] used multiple UAVs to serve as MEC/fog nodes for the provision of computing offloading services to ground mobile/IoT devices. For example, the study in [26] aimed to achieve load balancing while still maintaining the coverage constraint and service satisfaction, while that in [27] minimized the energy consumption of the ground devices by jointly optimizing the UAV task scheduling process, bit allocation, and trajectory. The study in [28] used multiple UAVs to construct a fog computing architecture and formulated the task allocation problem as that of minimizing the UAV energy consumption under the joint constraints of latency and reliability.

**Cellular-enabled UAV research.** Enabling cellular connectivity for UAV communications has attracted growing attention in industry and academia in recent years [4]. Many industrial institutions, including Qualcomm and AT&T, have expended significant effort in optimizing 4G/5G networks for UAV communications [5]. Lin et al. [29] considered the problem of enabling LTE connectivity for low-altitude small UAVs and presented several strategies for improving the LTE performance. Zeng et al. [8] surveyed the many challenges and opportunities of LTE-enabled UAVs. Khamidehi et al. [30] considered the trajectory optimization problem for multi-UAV-enabled multi-UAV-enabled cellular networks, in which the UAVs carried cellular BSs, and attempted to maximize the minimum data rate of the ground mobile users under the joint constraints of the UAV power, backhaul link capacity, and need for collision avoidance.

However, the literature lacks any study on control delay modeling over wireless dynamics for edge-based UAVs in cellular networks. This paper commences by considering the control delay behaviors of two practical MEC testbeds. The empirical delay data are then used for reference purposes in deriving an analytical model for estimating the expected error between consecutive control delays in order to improve the ability of the GCS to predict the location of the UAV and to formulate its control decision accordingly.

III. EMPIRICAL CONTROL DELAYS FOR EDGE-ENABLED UAV

This section examines the practical communication delays of the UAV control messages on two MEC platforms, namely a Bump-in-the-Wire platform and a Distributed S-GW with Local Breakout platform. (Note that both platforms adopt LTE network architectures and are standard-compliant solutions without new functions deployed in the 3GPP components.) The discussions commence by describing the detailed architectures of the two platforms implemented in the present study. The corresponding experimental settings and empirical results are then briefly introduced and discussed.

**A. MEC Platforms**

The first MEC platform was a middlebox MEC developed by National Yang Ming Chiao Tung University (NYCU) in Taiwan [31]. The second platform was an intelligent MEC
(iMEC), built by the Industrial Technology Research Institute (ITRI) of Taiwan [32], and also deployed at NYCU. Both platforms serve as MEC testbeds for commercial service trials and were selected in the present study for two main reasons: (1) they use two easily-deployed, standard-compliant methods, which can be widely adopted by other MEC platforms; and (2) the iMEC platform is deployed in the commercial Ericsson cellular infrastructure, and can thus be regarded as a reference MEC solution for the cellular industry.

For each platform, edge-enabled UAV control was enabled by deploying the UAV GCS as an application server on the platform. The details of the deployment process for the two platforms are described in the following.

**1) Middlebox MEC Platform:** The middlebox MEC adopts the Bump-in-the-Wire deployment method and hence serves as a middlebox sitting on the S1-U/S1-MME interface next to the eNB, as shown in Figure 5. Notably, it requires no modifications to the E-UTRAN or EPC, and only needs to handle the GTP tunnels over the S1 interface. This experimental testbed was deployed with an open-source LTE core, NextEPC [33], together with a commercial LTE small cell for the eNB.

In general, the middlebox platform provides MEC functionality via three major components. First, a GTP traffic engineering module encapsulates and decapsulates the GTP packets and steers the packets to either the MEC cloud or the E-UTRAN/EPC. The module additionally forwards MEC traffic to/from the application servers at the MEC cloud from/to the GTP tunnels, respectively, while also performing the usual message exchanges between the E-UTRAN and EPC in the control and data planes. Second, the platform uses the DNS service to redirect Internet traffic to the MEC cloud. Specifically, it returns local IP addresses to DNS requests with the domain names which have their application servers deployed at the edge. Third, the platform holds multiple applications in containers and orchestrates them using OpenStack. In the present experiments, UAV GCS was deployed at a container inside the cloud.

**2) iMEC MEC Platform:** The iMEC uses the Distributed S-GW with Local Breakout approach, which builds an S-GW next to the eNB in addition to the one in the EPC (see Figure 6). The local S-GW connects to the P-GW and eNB via S5 and S1-U interfaces, respectively. The platform requires the MME to perform S-GW selection such that the UEs are served by the local S-GW rather than the EPC S-GW. Therefore, the EPC needs to be aware of the MEC support with small operational updates. This platform was integrated with the Ericsson EPC and eNB platforms.

In general, the iMEC platform retains the original S-GW functions and builds an MEC platform on top of them. It provides a complete orchestration and deployment management service for the MEC application servers. Specifically, it supports automatic deployment and removal of the servers, control and monitoring of their life cycles, installment of their required traffic rules, and so on. iMEC consists of two main components in addition to the DNS service. First, it has an MEC management system that supports all automatic operations and provides users with a user interface to the MEC management and configuration tasks. Second, it supports two virtualization types for the MEC application servers: (1) container-based network function virtualization (NFV) with Kubernetes [34]; and (2) virtual machine (VM)-based NFV with Open Platform NFV (OPNFV) [35]. In the present study, the UAV GCS was deployed at a container on Kubernetes.

**B. Experiment Settings**

For the middlebox MEC platform, an LTE network was constructed consisting of a PC for the EPC, a commercial small cell for the eNB, and a server for the MEC. For the iMEC platform, the LTE network integrated with the iMEC was leased from ITRI. For both platforms, an embedded

---

**Fig. 5:** Middlebox MEC platform based on Bump-in-the-Wire method.

**Fig. 6:** Ericsson iMEC platform based on the Distributed S-GW with Local Breakout method.

**Fig. 7:** DJI Matrice 100 UAV.
TABLE I: Hardware equipment and software of middlebox and iMEC MEC platforms.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Equipment and Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Middlebox MEC</td>
<td>Intel Xeon E5-2680 (16GB RAM)</td>
</tr>
<tr>
<td></td>
<td>ERICSSON LTE EPC</td>
</tr>
<tr>
<td></td>
<td>Intel Core i7-7700 (16GB RAM)</td>
</tr>
<tr>
<td></td>
<td>WNC OSQ4G-01E2</td>
</tr>
<tr>
<td></td>
<td>Huawei E3372h LTE dongle</td>
</tr>
<tr>
<td>Internet Cloud</td>
<td>Intel Xeon E5-2680 (16GB RAM)</td>
</tr>
<tr>
<td></td>
<td>ERICSSON LTE EPC</td>
</tr>
<tr>
<td></td>
<td>Intel Core i7-7700 (16GB RAM)</td>
</tr>
<tr>
<td></td>
<td>Dell Poweredge R630</td>
</tr>
<tr>
<td></td>
<td>Intel Atom x5-z8350 (2GB RAM)</td>
</tr>
<tr>
<td></td>
<td>Lanner NC4210</td>
</tr>
<tr>
<td></td>
<td>Intel Xeon E5-2650</td>
</tr>
<tr>
<td></td>
<td>Intel Xeon E5-2650 (16GB RAM)</td>
</tr>
<tr>
<td></td>
<td>NextEPC (Ubuntu 18.04 LTS)</td>
</tr>
<tr>
<td></td>
<td>WNC OSQ4G-01E2</td>
</tr>
<tr>
<td></td>
<td>Huawei E3372h LTE dongle</td>
</tr>
<tr>
<td>iMEC MEC</td>
<td>Intel Atom x5-z8350 (2GB RAM)</td>
</tr>
<tr>
<td></td>
<td>Intel Xeon E5-2650 (16GB RAM)</td>
</tr>
<tr>
<td></td>
<td>NextEPC (Ubuntu 18.04 LTS)</td>
</tr>
<tr>
<td></td>
<td>WNC OSQ4G-01E2</td>
</tr>
<tr>
<td></td>
<td>Huawei E3372h LTE dongle</td>
</tr>
</tbody>
</table>

Intel x86 platform (AAEON UP-Board) was used as a UE to connect to the LTE network through an LTE dongle. As shown in Figure 7, the board was mounted directly on the UAV. Table I summarizes the hardware and software details of the two platforms.

In performing the experiments, the UE sent a 1 kB packet to the GCS, and the GCS replied with a control packet as soon as it received it. For each platform, the delay (i.e., RTT) was measured as the time between the moment at which the UE sent out the video frame and that at which it received the control packet. The RTT was measured for three different signal strengths at the UAV: strong, medium, and weak. Furthermore, for each MEC platform and every signal strength, measurements were obtained for two different GCS deployments, namely at the MEC and in the Internet cloud. For every experiment, 20000 RTT samples were collected, with an interval of 10 ms between them in every case.

C. Measured UAV Control Delays

Figure 8 shows the measured control delays of the UAVs for the middlebox MEC platform and cloud implementations, respectively, given the three different signal strengths. Figure 9 presents the corresponding results for the iMEC platform. For a strong signal at the UAV (Figures 8a and 9a), the RTTs of the UAVs are notably shorter for the MEC deployments of the GCS than for the cloud deployment. For example, in terms of the 50th and 90th percentiles, the middlebox MEC RTTs are 29.3 ms and 30.8 ms, respectively, and are around 23.3% and 33.2% quicker than those of the cloud RTTs (Figure 8a). Similarly, for the iMEC MEC, the corresponding RTT values are 15.5 ms and 22.1 ms, respectively, and are around 26.5% and 20.8% faster than those of the cloud. Comparing the two figures, it is found that the RTT values of the Ericsson eNB are around 38.3% lower than those of the WNC small cell.

For a medium signal strength (Figures 8b and 9b), the middlebox MEC RTT values are still lower than those of the cloud; however, the difference is less appreciable than in the case of a high signal strength. For the iMEC platform, the performance of the Ericsson eNB dominates the RTT value, and hence the RTT values are similar for both GCS deployment strategies. Finally, for a weak signal strength (Figures 8c and 9c), the MEC gains are offset by the poor wireless performance, and hence the RTT values of both MEC platforms are similar to those of the cloud.

IV. MODELING-BASED RANGE ESTIMATION ON UAV CONTROL DELAY

This section commences by describing the UAV control delay estimation problem considered in the present study. The modeling-based solution proposed for estimating the error...
range of the control delay is then formulated. Finally, the application of the proposed modeling approach to the practical UAV operation is then briefly described.

A. Problem Statement

It is assumed that the UAV periodically reports its sensed data to the GCS through the CNPC link, and the GCS acknowledges the receipt of this data by sending back a packet, which may or may not contain control commands. The UAV measures the control delay between itself and the GCS for each round trip performed in this way, and then feeds this delay value back to the GCS in its next report. The GCS collects all the control delay data over time and uses this information to predict the upcoming delay. As shown in Figure 2, every time the GCS receives sensed data sent from the UAV at time $T_0$, it estimates a range of possible control delays $t' = T_2 - T_0$ at time $T_1$, and then uses this estimated range to predict the possible location range of the UAV at time $T_2$ as the product of the estimated control delay range and the UAV speed.

As described above, the GCS estimates the value range of each control delay based on a knowledge of the historical delay data. In practice, the control delay includes not only the RTT between the UAV and the GCS, but also the processing times of the sensed data and queuing times of the data packets at the two communication ends, as well as the packet transmission time. However, for simplicity, this section considers the empirical control delays obtained in Section III for illustration purposes. These delays refer to only minimal sensed data (e.g., only GPS and some inertial data). Hence, the processing and transmission times are both very small. Moreover, UAV traffic is regarded as high priority traffic by the cellular network, and consequently, the queuing time is also very small. As a result, the delay time is dominated by the RTT, and the other delay components are thus simply ignored. However, it is noted that the general modeling approach described in this section is expected to be equally applicable to other types of control delays, including those dominated by other factors, such as long processing times for image encoding and decoding.

B. Modeling-based Range Estimation

In the present study, the control delay is modeled by approximating a continuous probability distribution to the historical delay data over time. Suppose that the next and previous control delays are denoted as $t'$ and $t$, respectively, and the error between two consecutive delays is defined as $\tau = |t' - t|$, where $\tau \geq 0$. The expected difference (i.e., error) between two consecutive delays, namely $E[\tau]$, can then be derived from the approximated probability distribution and used to estimate the value range of the next control delay as $[t - E[\tau], t + E[\tau]]$.

The control delays are assumed to be i.i.d. random variables with a density function $f()$. The density function can be derived by approximating the histogram of the measured delays. The expected value of $\tau$ can then be formulated as:

$$E[\tau] = \int_{t=0}^{\infty} \int_{t'=0}^{\infty} \tau f(t)f(t')dt'dt,$$  \hspace{1cm} (1)

For the purposes of the present study, the control delays are assumed to come from an Erlang distribution, where the shape and scale parameters of this distribution can be adjusted as required to approximate the histogram of the measured delays. Under stable network conditions, the delay distribution has a unimodal form; however, under more realistic conditions, an interplay inevitably exists between multiple delay factors, and hence the control delays are more reasonably modeled as a bimodal distribution (see Figures 8 and 9). Note that other distributions could also be considered for approximation purposes, provided that the approximated distributions fit the measured histogram well.

The expected errors between consecutive delays, $E[\tau]$, for the unimodal and bimodal distribution approximations of the control delays can be derived as follows.

1) Control Delays with Unimodal Distribution: Without loss of generality, let $t' = t + \tau$ and $t = t' - \tau$ when $t' \geq t$ and $t' < t$, respectively. Furthermore, assume that $t$ and $t'$ are Erlang-distributed random variables such that

$$f(t) = \frac{\lambda^k e^{-\lambda t}}{(k-1)!},$$ \hspace{1cm} (2)

where $k \in 1, 2, 3, \ldots$ and $\lambda \geq 0$.

Then, Equation 1 can then be rewritten as:

$$E[\tau] = \int_{t=0}^{\infty} \int_{t'=0}^{\infty} \tau f(t)f(t + \tau)d\tau dt + \int_{t'=0}^{\infty} \int_{t=0}^{\infty} \tau f(t')f(t' + \tau)d\tau dt',$$ \hspace{1cm} (3)

where the two terms on the right-hand side are equal since the two random variables, $t$ and $t'$, have the same distribution. As shown in Appendix A, Equation 3 can be rewritten as

$$E[\tau] = 2 \sum_{j=0}^{k-1} \frac{(2k - j - 2)!}{(k - 1)!} \left( \frac{k+1}{2k-j-1\lambda} \right).$$ \hspace{1cm} (4)

For the particular case of $k = 1, t$ and $t'$ are both exponential random variables. So, Equation 4 can be simplified as

$$E[\tau] = \frac{1}{\lambda}. \hspace{1cm} (5)$$

2) Control Delays with Bimodal Distribution: Observing the control delays collected from the middlebox MEC and iMEC MEC platforms in Section III, it is seen that the delay distributions for both platforms have a bimodal characteristic given a strong signal at the UAV (see Figures 8a and 9a). Moreover, the delay distribution for the middlebox MEC platform also has a bimodal characteristic given at medium strength signal at the UAV (see Figure 8b). As described in Section III, in the present study, the bimodal distribution of the control delays is modeled as a mixture of two Erlang distributions, $f_1(t)$ and $f_2(t)$, with random variables $t_1$ and $t_2$, respectively, i.e.,
operations, just before sending its sensed data to the edge
two consecutive control delays, to approximate the corresponding continuous probability dis-
time, the modeling approach described above can be used
agreement with the experimental histogram. Hence, the valid-
As shown in Figure 10, the resulting approximated bimodal
parameters of
parameters
parameters
E(15)−E(0)=[t−mE[τ]+mE[τ]], where t is the previous control delay and m is a coefficient as a multiple of E[τ]. It then sends this
delay range, together with the sensed data, to the edge server. Given a knowledge of the average speed of the UAV, \( \bar{v}_m \), the GCS at the edge server uses the delay information to predict the probable movement distance of the UAV in the upcoming
control delay \( t' \) as \( \bar{v}_m(t−mE[τ]), \bar{v}_m(t+mE[τ]) \).

V. EVALUATION

An event-driven simulator was derived to observe the relationship between the prediction accuracy of the control delays (as evaluated using the modeling approach described above) and the expected error between consecutive delays, \( E[τ] \). The simulator was further used to examine the effects of the bimodal distribution parameters on the magnitude of \( E[τ] \). The analysis focused on the middlebox and iMEC platforms with a strong signal strength at the UAV. For both platforms, the simulator was used to generate a series of control delays from the corresponding bimodal distribution approximations of the RTT measurement reported in Section III in accordance with the procedure shown in Appendix B.

A. Validation of Event-driven Simulator

The validity of the event-driven simulator was evaluated using the analytical model derived in Section IV-B2 for the bimodal distribution of the control delay time. In particular, parameters \( k_1, λ_1, k_2, \) or \( λ_2 \) in Equation 6 were separately varied while fixing the other parameters. In each case, the error was calculated for all possible sets of two consecutive delays (i.e., \( τ \)), and the expected delay error, \( E[τ] \), was taken as the simulation result. Figure 11 compares the analytical and simulation results for the variation of \( E[τ] \) with \( λ_1 \) and \( k_1 \), respectively. It is clear that the two sets of results are in excellent agreement with one another for both parameters. Overall, the average discrepancy between the analytical and simulation results over all the considered parameter settings was found to be less than 0.5%. Thus, the validity of the event-driven simulator was confirmed.

B. Prediction Accuracy

The bimodal distribution parameters, \( π, k_1, λ_1, k_2, \) and \( λ_2 \), were estimated for both MEC platforms (middlebox and
iMEC) under strong signal conditions. The event-based simulator was then used to generate a series of control delays for each platform. For each control delay $t$, the next delay $t'$ was evaluated analytically in the range of $[t - mE[τ], t + mE[τ]]$, where the expected error, $E[τ]$, was derived from the approximated parameters and $m$ was assigned in the range of 0-3. The prediction accuracy of the analytical model was then evaluated by confirming whether or not the corresponding simulated delay value fell within the estimated range. The overall prediction accuracy of the model was finally obtained by averaging the prediction accuracy over all pairs of consecutive delays.

Figure 12a shows the variation of the prediction accuracy with the value of $m$ (i.e., a coefficient describing the multiples of $E[τ]$ considered when estimating the next control delay). As expected, the prediction accuracy increases with increasing $m$ due to the corresponding increase in the control delay prediction range. For the middlebox MEC, the prediction accuracy increases from 95% to 99% when $m$ increases from 2.2 to 2.5. It is noted that when a larger $m$ results in a higher prediction accuracy for the control delay, the resulting broadening of the UAV location prediction range may cause the GCS to make more conservative decisions in order to prevent collisions.

C. Relationship between $E[τ]$ and $π$

The effect of parameter $π$, i.e., the mixture coefficient of the bimodal distribution, on the expected error between consecutive delays, $E[τ]$, was investigated by varying $π$ in the range of 0-1 in intervals of 0.1 while fixing all the other distribution parameters, and generating a series of control delays for each case using the validated simulator. Figure 12b presents the corresponding results for the two MEC platforms. It is seen that $E[τ]$ varies as a concave-downward function of $π$ in both cases. In other words, for $π = 0$ or 1, the bimodal distribution of the control delays reverts to a unimodal distribution. For the unimodal distribution, the control delay data are more densely concentrated, and hence the expected error of consecutive delays, $E[τ]$, is smaller. By contrast, the largest value of $E[τ]$ occurs at $π = 0.5$, for which the two unimodal distributions within the bimodal distribution have an equal share. For the middlebox MEC, $E[τ]$ has values of 1.83, 1.08 and 5.84 for $π = 0$, 1 and 0.5, respectively.

Overall, the results shown that to minimize the value of $E[τ]$, and thus narrow the prediction range of the UAV future location, the network provider should aim to stabilize the delays of the LTE network and MEC platform, respectively. In the ideal case, the control delays should be distributed with a unimodal distribution in order to achieve the smallest possible value of $E[τ]$.  

D. Relationship between $E[τ]$ and Mean Distance between Mode Means

The effects of the diversity of the bimodal distribution on the expected error between consecutive delays, $E[τ]$, was evaluated by varying the distance between the means of the two unimodal distributions in the bimodal distribution while retaining all the other distribution parameters unchanged. As shown in Figure 12c, $E[τ]$ increases with an increasing separation of the means for both platforms. For the case where the two unimodal distributions have the same mean (i.e., the bimodal distribution has the form of a single unimodal distribution), the control delay data are most densely concentrated, and hence $E[τ]$ has its minimal value. Conversely, as the distance between the two means increases, the control delays are more diversely distributed, and hence each control delay serves as a less reliable predictor of the following delay. Thus, the results indicate that, if a bimodal distribution of the control delays cannot be avoided, the network provider should aim to minimize the separation distance of the two means as much as possible in order to reduce the expected error between consecutive delays and hence improve the ability of the GCS to more accurately predict the expected location of the UAV at the moment it receives the upcoming control command.

VI. CONCLUSION

Cellular networks have the potential to equip UAV wireless control systems with not only a long-range communication capability, but also a low latency performance through the use of MEC technology. This study commenced by measuring the RTT between a UAV and an edge-based GCS on LTE platforms with two different MEC deployment methods (middlebox and iMEC). The results confirmed that the edge-based implementation of the GCS yielded an effective reduction in the mean RTT of the UAV-GCS system compared to that for a traditional cloud-based implementation. However, even though
the MEC-based deployment of the GCS reduces the control delay, obtaining precise estimates of each control delay is still essential in enabling the GCS to formulate appropriate control commands. Accordingly, a modeling approach has been proposed for approximating the historical control delays within the UAV-GCS system as a bimodal distribution. Given this distribution, the expected error range between consecutive control delays can be estimated by the UAV and then used by the GCS to predict the probable location range of the UAV. The validity of the proposed modeling approach has been confirmed using an event-based simulator. The results have shown that the ability of the GCS to predict the future position of the UAV can be improved by stabilizing the control delays within the UAV and MEC-enabled system. Overall, the results presented in this study provide a useful contribution to ongoing research on edge-enabled UAV control.

**Appendix A**

**Derivation of $E[\tau]$**

For the unimodal distribution described in Section IV-B1, the expected error between consecutive control delays (Equation 3) can be derived as follows:

$$E[\tau] = 2 \int_{\tau=0}^{\infty} \left[ \lambda t e^{-\lambda t} \right] dt = \frac{2}{\lambda}$$


[12] 3GPP, General Packet Radio Service (GPRS); GPRS Tunneling Protocol (GTP) across the Gn and Gp interface. 3GPP Standard TS29.060 V15.5.0, 2019.


