Optimizing Quality of Experience for Long-Range UAS Video Streaming

Russell Shirey, Sanjay Rao, Shreyas Sundaram
School of Electrical and Computer Engineering, Purdue University
{rshirey, sanjay, sundara2}@purdue.edu

***Abstract*—There is much emerging interest in operating Unmanned Aerial Systems (UAS) at long-range distances. Unfortunately, it is unclear whether network connectivity at these distances is sufficient to enable applications with stringent performance needs. In this paper, we consider this question in the context of video streaming, an important UAS use-case. We make three contributions. First, we characterize network data collected from real-world UAS flight tests. Our results show that while dropouts (i.e., extended periods of poor performance) present challenges, there is potential to enable video streaming with modest delays, and correlation of throughput with flight path (both distance and orientation) provides new opportunities. Second, we present Proteus, the first system for video streaming targeted at long-range UAS settings. Proteus is distinguished from Adaptive Bit Rate (ABR) algorithms developed for Internet settings by explicitly accounting for dropouts, and leveraging flight path information. Third, through experiments with real-world flight traces on an emulation test-bed, we show that at distances of around 4 miles, Proteus reduces the fraction of a viewing session encountering rebuffering from 14.33% to 1.57%, while also significantly improving well-accepted composite video delivery metrics. Overall, our results show promise for enabling video streaming with dynamic UAS networks at long-range distances.**

***Index Terms*—Unmanned Aerial Systems (UAS), Adaptive Bit Rate (ABR) Algorithms, Video Streaming, Networking, Drones.**

I. Introduction

Recent technological advances have dramatically increased the availability and capabilities of Unmanned Aerial Systems (UAS) [1]. Once limited to a small community of research and military applications, the barrier to entry for owning and operating UAS, sometimes referred to as drones, has decreased in recent years. UAS are used in many domains including military [2], [3], disaster response [4]–[6], search and rescue [7], law enforcement [4], agriculture [4], railroad and pipeline inspection [4], [8], and more [4], [9], [10].

Many UAS applications involve recording and streaming video. Quality and reliability (e.g., uninterrupted video) is important when the video is being viewed by a human (e.g., to monitor and take appropriate action). Since UAS must travel to locations determined by the application requirements (e.g., surveying areas that are dangerous or difficult to access by humans such as disaster-hit areas, and military environments), there is limited freedom in placing UAS to optimize for connectivity. This is in contrast to the use of UAS to extend Internet connectivity to remote locations [11], [12], where it is feasible to optimally position UAS to ensure the best connectivity. Today, civilian use of UAS in the United States (US) is typically limited to distances that require visual line of sight [13] (typically, 1 Km). However, there is interest in extending the range of UAS networking, and regulations are being put into place to support such initiatives [1], [8], [9].

In this paper, we tackle two key questions: (i) What are the characteristics of long-range UAS networking settings, and what challenges do they pose for video streaming? (ii) How should video streaming algorithms be designed to address these challenges, and is it viable to achieve good performance? The answers to these questions are not obvious *a priori*, and are complicated by the lack of real-world flight data.

Contributions. In this paper, we answer the above questions and make the following contributions.

First, we provide an analysis of UAS networks based on real-world data from UAS flights at long-range distances exceeding traditional civilian regulations (through special approval). Our measurements are obtained from flying fixed wing UAS, which have higher speed and longer endurance than multirotor systems [14]. The results show that ultra-low latency (e.g., sub-second) video streaming may not be achievable, due to the prevalence of dropouts (periods of extremely poor throughput). However, the typical dropout duration is short enough that video streaming with delays of tens of seconds is potentially viable, given the right algorithm design. This is an acceptable delay in many of the UAS applications we described, especially if the result extends the usable flight distance. Further, our study shows that wireless performance depends not only on distance but also, more interestingly, on UAS orientation.

Second, motivated by the observations above, we present Proteus, a system for video streaming in UAS settings. Proteus leverages Adaptive Bit Rate (ABR) algorithms [15]–[20], given they are well suited to streaming with modest delays, and since the throughput that can be sustained in UAS settings is highly variable and dependent on flight path. While ABR algorithms have been extensively studied in traditional Internet environments, they are only starting to be explored in UAS settings [10], [21]. To our knowledge, Proteus is the first system for video streaming that tackles long-range UAS settings, and issues unique to fixed-wing UAS.

Proteus is based on a control-theoretic approach, motivated by the success of such approaches for traditional Internet video

978-0-7381-3207-5/21/$31.00 ©2021 IEEE

streaming [16], [19], [22]. Unfortunately, we illustrate that a direct application of a representative and widely studied ABR algorithm based on a control theoretic approach [16], often referred to as MPC in the video streaming community[1], does not work well for long-range UAS environments. Indeed, we show that even with a *perfect predictor* of throughput (i.e., an Oracle), MPC performs poorly owing to extended dropouts beyond the finite planning horizons utilized by the algorithm.

Proteus mitigates such myopic decision-making by introducing a *terminal cost* into the receding-horizon optimization at each point in time. Such terminal costs are commonly used in control theory as a way to introduce long-term considerations into short-term planning [23]; however, an open problem is how to *choose* the terminal cost appropriately for each individual problem. In our setting, we show that by carefully constructing a terminal cost that incentivizes increasing buffer occupancy to hedge against future dropouts, we can obtain substantial gains in video streaming performance over long-range UAS networks. In particular, we show that knowledge of the UAS flight path, which is typically known ahead of time because of regulations [13] and mission planning requirements, can be incorporated into the design of the terminal cost by choosing the parameters as a function of *both* the UAS distance and orientation (motivated by our analysis of UAS network characteristics from our first contribution).

Third, we show through experiments, using real-world network traces from UAS flights on an emulated test-bed, that Proteus significantly improves performance compared to MPC. For example, we reduce the rebuffering ratio from an average of 14.33% to 1.57% for a flight trace flying a circle orbit around a point 4 miles away from a receiver on the ground, while also significantly improving a well accepted composite metric, Quality of Experience (QoE) [16]–[18], for video delivery. Overall, these results show the promise of enabling video streaming applications over variable UAS network environments at long-range distances with Proteus.

II. Motivating measurements of UAS networks

In this section, we present measurements from real-world UAS flights. Our data reveals the challenges of UAS video streaming over distances stretching the limits of wireless connectivity due to dropouts and highly varying throughput. The results also provides insights on how network performance correlates with the UAS flight path, which we leverage in our video streaming algorithm design in later sections.

A. UAS flight test setup

We flew a representative fixed wing UAS in terms of size, weight, and speed, and appropriate for the activities described in §I [14]. There are two broad kinds of UAS - fixed wing and multirotor systems [14]. Fixed wing systems are similar to traditional aircraft, with a central body and two wings. Multirotor systems are similar to a helicopter structure, with four (quad) or more rotors. In this paper, we focus on fixed wing systems, since the video streaming applications we focus on benefit from speed and longer endurance.

To better understand and describe our flight data, we introduce the common flight terms **Distance, Slant Range, and Altitude**, relative to the Ground Control Station (GCS), and show them in Fig. 1. The UAS orientation of **coming towards** (to the GCS) and **going away** (from the GCS) are also shown.

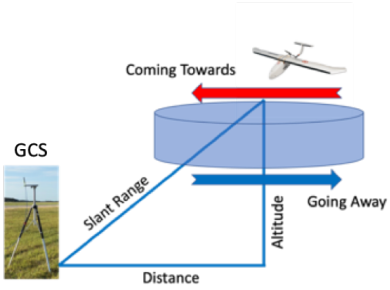

Figure 1. UAS terms and orientation

Circular orbits around a point are useful for recurring video coverage of an area. We refer to the data collected from circle orbits as **circ(k)**, with k = {1, 2, 3, or 4}, depending on the distance from the GCS to the center of the circle. We also flew along a line pattern (we refer to the collected data as the **Line** trace), which is useful for monitoring of multiple successive areas (e.g., pipeline inspection).

For UAS wireless connectivity, we used two Persistent Systems MPU4 Tactical Radios [24], [25] tuned to S-Band frequency and in a point-to-point configuration (one on the UAS and the other on the ground). These radios provide layer 2 connectivity and support Internet Protocol (IP) traffic. We tested with an omnidirectional antenna on the UAS and tested separately with both directional and omnidirectional antennas on the ground. We focus on the omnidirectional ground antennas since they are most common and best for application flexibility (a tracker is not needed since they do not have to point at the UAS). This enables the antenna to be mounted almost anywhere, even on a person, and moved around at ease. Directional antennas are sometimes still used in practice [26], and we also provide test results for directional antennas in §IV in order to show the flexibility and broad applicability of our work. We collected TCP throughput using iPerf [27] and used pings to record latency and loss rate.

B. Findings and implications for UAS video streaming

Fig. 2 shows time series plots of the UAS slant range and network metrics (throughput, latency, and SNR) for the circ(4) trace. The figures show network performance, and how it varies with slant range. While latency does not have a distinct pattern, dropouts are more prevalent at extended distances, especially during the *coming towards* duration (from about 50 until 140 seconds). Dropouts are sections where the throughput is 0 (top-left plot) and pings are lost (bottom-left plot). This qualitatively suggests that UAS orientation will have an impact

[1]While Model Predictive Control refers to a broad body of work in the control literature, we use MPC more narrowly to refer to a specific streaming algorithm [16], following convention in the video streaming community

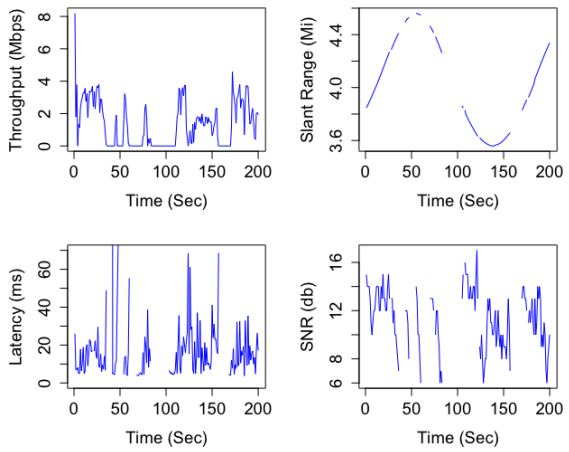

Figure 2. Network metrics for circ(4) flight

on network performance. We will see that this impact is indeed present, and will quantify the differences in the subsequent subsections. The other traces were qualitatively similar. We now discuss two salient features of our measurements, along with their implications for video streaming.

Losses and dropouts impact latencies achievable with video streaming. We measured loss rate, using pings, of 5.6%, 18.3%, 28.5%, and 23.5%, respectively for circ(1-4). The loss rate increases with distance, except a minor decrease for circ(3) to circ(4); this anomaly is likely due to the UAS spending more time in the lower performance *coming towards* orientation during circ(3), compared to circ(4).

We next measure consecutive periods of loss, which is an indication of no data going through for a certain period. Fig. 3 shows the median and maximum duration of consecutive losses, measured with pings. The median duration increases slightly with distance and the maximum value increases from circ(1) to circ(3), and slightly decreases for circ(4).

We next look at TCP dropout periods and compare performance by distance and orientation. We define a **dropout period** as at least one second in which the throughput is zero. Dropout periods are often more prevalent at extended distances and are typically caused by the SNR dropping below a certain threshold, which causes a temporary wireless link failure.

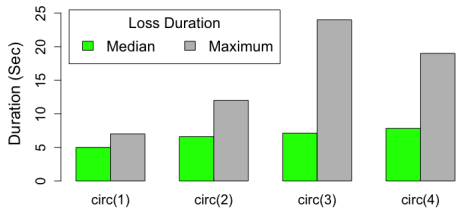

Figure 3. Median and maximum loss duration with pings

Fig. 4 shows the average duration of dropouts for each orbit, as well as for the *coming towards* and *going away* phases of the orbits. First, we observe that the average dropout duration generally increases as the distance of the plane increases, both for the orbit as a whole, and during each phase of the orbit. Second, the average dropout duration is generally higher in the *coming towards* phase of each orbit than in the *going away* phase. The data in Fig. 4 does show some exceptions to the above trends, namely when going from circ(2) to circ(3) (for the *going away* phase), and from circ(3) to circ(4) (for the *coming towards* phase). These exceptions are due to two short dropouts of 1 second each in the corresponding traces.

These extended dropouts indicate that achieving extremely low latency video streaming with sub-second delays may not be viable without losing entire periods of video. Therefore, we focus on streaming with delays of tens of seconds, which our data suggests may still be potentially viable. This is still acceptable in many application scenarios as we discuss in §III.

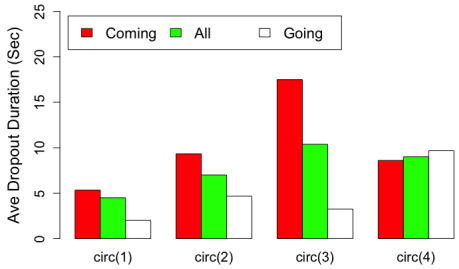

Figure 4. Average dropout duration

Correlation of throughput with flight path presents opportunities to improve video streaming. Fig. 5 plots the TCP throughput measured across the traces. Each group of boxplots corresponds to a trace, and shows the distribution of throughput samples collected at 1 second intervals for the entire trace (*All*), and the portions of the trace corresponding to the *coming towards*, and the *going away* orientations.

We see that throughput is generally higher for the *going away* orientation, compared to *coming towards*, consistent with earlier measurements. We believe this is because fixed wing UAS are not symmetrical and parts of the aircraft can

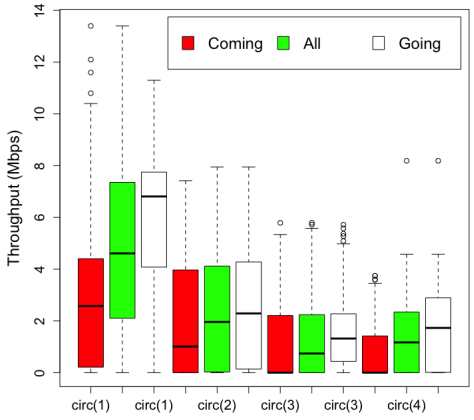

Figure 5. Throughput analysis with distance and orientation

interfere with wireless communication. This is common in fixed wing aircraft and optimal antenna placement depends on the intended environment, mission sets, and aircraft flight patterns. Additionally, a closer look at the representative radiation patterns on the specification sheet [28] for the UAS antenna shows weaker signal strength in the front of the antenna compared to the rear (by 1.5-2 dB). **Both the fixed wing UAS structure and antenna pattern** contribute to UAS orientation performance differences. Specifically, there exists lower network performance in the *coming towards* phase of the orbit, consistent with our data analysis.

Further, as expected, throughput generally decreases as distance increases. There is an exception to the trend as we move from circ(3) to circ(4), likely caused by the fact that circ(3) has additional time in the *coming towards* orientation, whereas circ(4) has additional time in the *going away* orientation, resulting in fewer dropouts. In §III, we will explore how Proteus leverages these correlations in its design.

III. Proteus: Video streaming in dynamic UAS environments

In this section, we present Proteus, a system for video streaming at long-range UAS settings, motivated by the measurements presented in §II.

A. Proteus design rationale

Solutions for video streaming in UAS settings have multiple design points. At one end of the spectrum, one can target ultra-low sub-second latencies. However, our measurements in §II indicate that this design is likely to incur significant video content loss owing to extended dropouts at these distances. At the other extreme, video may be recorded during the flight, and transmitted later to the ground [29]. This approach ensures high video quality, but unacceptable delays of tens of minutes, or even hours. Proteus targets a middle ground between video quality and latency, targeting latency in the range of tens of seconds, and choosing to degrade video quality when needed.

While Proteus cannot handle some applications that require near instantaneous decision making (e.g., military operations with troops actively engaged), delays of tens of seconds can be acceptable in military and civilian applications [2], [29]–[31], especially when the delay extends the mission flight range. For instance, in disaster response settings [4], [6], the extended distance can be very beneficial for safety and surveillance, and information received within tens of seconds can guide decisions such as when and where to deploy personnel to help. Further, such information can guide decisions on where to next fly the UAS. The primary benefit of Proteus is significant range extension of the UAS through software, while supporting the timeliness requirements of a vast majority of applications, and not requiring more expensive hardware solutions.

Proteus targets settings for human end-users, where stalls and fluctuations in quality are undesirable. Given these requirements, Proteus considers video delivery using ABR algorithms, which are a natural fit and can take advantage of varying network performance while optimizing bitrate quality. ABR algorithms can run over TCP or emerging approaches, such as QUIC [32], which allows use of rate adaptation mechanisms on top of UDP. We next discuss why existing ABR algorithms are inadequate, and present Proteus' approach.

B. Need for new ABR approaches for UAS settings

While many ABR algorithms have been developed in recent years [15]–[20], [33], they are all designed for traditional Internet environments. This poses a problem when working in dynamic UAS environments, where dropouts are more common and can cause rebuffering.

We focus on MPC [16], a widely used and representative ABR algorithm. Our insights can benefit other ABR algorithms as well, as we discuss in §V. MPC uses a combination of future throughput prediction and buffer occupancy to select chunk bitrates [16].[2] Prior to downloading each chunk, the MPC algorithm selects the next bitrate by optimizing a composite metric (more formally defined in §III-C) that rewards higher chunk bitrates, and penalizes rebuffering and variation of bitrates over a look-ahead of W future chunks. This type of scheme is also referred to as receding horizon optimization in the control literature [23].

We tested MPC via an emulated test-bed (see §IV for experimental setup details) with throughput traces gathered from our UAS flights. Fig. 6 shows the rebuffering ratio for video streaming using MPC with our flight test traces ranging from circ(1) to circ(4). For each trace, we present results for (i) the default harmonic mean throughput predictor, which for each chunk predicts throughput based on the harmonic mean of the throughput experienced by prior chunks; and (ii) a perfect Oracle predictor that provides exact throughput information for the duration of the look-ahead window. Rebuffering rates are high with the harmonic predictor, exceeding 25% for circ(3). Interestingly, while using the Oracle predictor helps, the rebuffering ratio is still high, exceeding 10% for circ(3). This is because while MPC optimizes bitrates for a given look-ahead, it can leave the buffer nearly empty owing to its greedy nature. This can in turn leave the algorithm vulnerable to dropouts and extended periods of low throughput beyond the finite planning horizons, which are quite common in long-range UAS settings. We experimented with both the default settings of MPC and settings with much higher weights for the rebuffering penalty, but found this inadequate to avoid rebuffering since this does not explicitly compensate for the greedy nature of the algorithm. We show more detailed results and comparisons with Proteus in §IV.

Next, we describe how to address these shortcomings by carefully incorporating characteristics of the flight path into the receding horizon optimization.

C. Proteus design details

In this section, we discuss our new algorithm design, Proteus, which overcomes previously discussed challenges by (i)

[2]We use the term MPC to refer to the conservative version of the algorithm (referred to as *RobustMPC* in [16]) which reduces predicted throughput by using a discount factor based on the maximum error in throughput predictions experienced over the last few chunks.

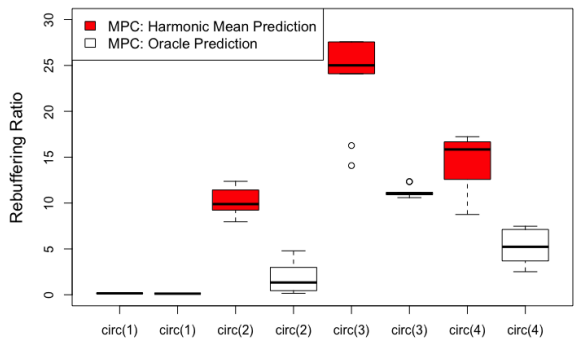

Figure 6. MPC rebuffering ratio for circ(4) with a harmonic mean predictor and an Oracle predictor

explicitly considering dropouts that occur in UAS networking environments, and (ii) incorporating knowledge of the flight path and its interplay with throughput. The improvements in Proteus are dependent on the addition of a *terminal cost* into the receding-horizon optimization at each point in time. Terminal costs are often used in receding horizon optimization to ensure stability and improve performance [23], however, choosing an appropriate terminal cost for each individual problem is a difficult and open problem. We discuss how we carefully design and select our terminal cost in order to substantially improve long-range UAS video streaming. Specifically, motivated by our network measurements and analysis, we show how we utilize knowledge of the UAS flight path in the terminal cost design by carefully choosing parameters that consider both UAS distance and orientation.

Handling dropouts: Consider that the overall objective of the ABR algorithm is to optimize a composite metric that rewards higher bitrates, penalizes rebuffering (and start-up latency), and penalizes variations in bitrates across chunks (smoothness). While a variety of composite metrics could be used with Proteus, we focus for concreteness on a metric obtained by a linear combination of these metrics as in past work [16]–[18], defined as:

$$QoE(i,j) = \sum_{n=i}^{j} R_n - \mu \sum_{n=i}^{j} T_n - \lambda \sum_{n=i}^{j-1} |(R_{n+1}) - (R_n)|. \quad (1)$$

Here, $QoE(i,j)$ captures the metric for chunks ranging from i to j. If N is the total number of the chunks in the video, the performance metric for the entire session is simply captured by $QoE(1,N)$, or denoted by QoE for brevity. R_n refers to the bitrate for chunk n (indexed relative to the current chunk) and T_n refers to the amount of rebuffering experienced. The coefficients μ and λ capture the extent to which rebuffering and bitrate variations are penalized.

Rather than greedily optimize the above metric in each finite planning horizon, Proteus mitigates the greedy nature of MPC by explicitly incentivizing that some amount of video is left in the buffer when selecting bit rates. Specifically, consider that a finite look-ahead window of W chunks is used, and chunks starting from chunk i are to be downloaded. Then, we create a new optimization metric for each look-ahead window (solved prior to the transmission of each new chunk i in a receding horizon fashion) as follows:

$$QoE_b = QoE(i, i+W-1) + \gamma\epsilon(b). \quad (2)$$

Note that the QoE_b metric optimized by Proteus in each look-ahead window includes a term $\gamma\epsilon(b)$ that considers the amount of video (b), in seconds, in the buffer at the end of the window. A higher value indicates that the algorithm wishes to insure more for the future. The function $\epsilon(b)$ ranges from 0 to 1 depending on the buffer occupancy b, while γ is decides how much weight to assign to the buffer insurance term, relative to other factors such as bitrate and rebuffering. In particular, by setting $\gamma = 0$, we obtain the original QoE metric from Eq. (1). The function $\gamma\epsilon(b)$ plays the role of a "terminal cost."

There are various considerations that go into the design of $\epsilon(b)$. A larger term indicates more insurance for the future, in case network performance degrades or a long dropout occurs, but also makes Proteus more conservative (since the algorithm may choose to sacrifice bitrate in order to fill up the buffer). While some insurance can help, the benefits diminish with larger insurance. Thus, the need to fill up the buffer to a "sweet spot" (in order to balance video quality and insurance against dropouts) motivates us to design an $\epsilon(b)$ that is quadratic in the buffer occupancy b. Specifically, we use the following:

$$\epsilon(b) = \frac{\bar{b}^2 - (\min(b, 2\bar{b}) - \bar{b})^2}{\bar{b}^2}, \quad (3)$$

where $\bar{b}$ represents a *target buffer size*, or desired level of buffer, in seconds, which provides some level of insurance against dropouts without being too conservative. The function $\epsilon(b)$ reaches a maximum of 1 when $b = \bar{b}$, but is 0 when $b = 0$, or $b \geq 2\bar{b}$. Fig. 7 shows a plot of $\epsilon(b)$ and the effect that changes in buffer occupancy has on insurance. We also explored a logarithmic function but found the quadratic to be slightly better and thus only discuss the latter in this paper.

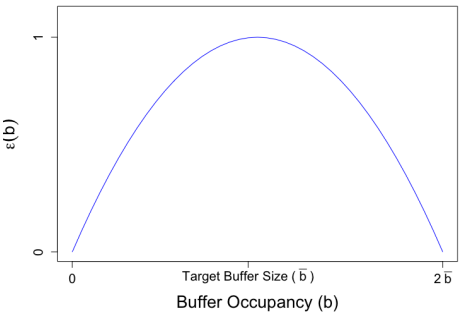

Figure 7. An illustration of the terminal cost $\epsilon(b)$ from Eq. 3, representing the reward obtained by having a buffer occupancy b.

Incorporating path awareness: In the scheme above, a key question is how to set the parameters $\bar{b}$ and γ. Intuitively, these parameters depend on the network characteristics (particularly, the duration of dropouts). As our measurements have shown,

the throughput primarily depends on the distance of the UAS from the GCS, and secondarily on the orientation. Based on these insights, we consider two schemes:

(i) **Proteus,** where the buffer insurance parameters are chosen based on the distance of the UAS from the GCS.

(ii) **Proteus-Orient,** where we select parameters based on both the distance of the UAS, and its orientation (*coming towards* and *going away*).

We set $\gamma = \alpha \times M \times W$, where M is the maximum bitrate and W is our video chunk look-ahead window. Here, α is used to regulate the importance of the buffer insurance term. When $\alpha = 1$, the maximum value of the buffer insurance term is equal to the maximum bitrate reward over the W-chunk look-ahead, while $\alpha = 0$ turns off buffer insurance.

We next discuss how the $\overline{b}$ and α parameters are set for Proteus in order to optimize performance across the flight path. For the circle traces, while the distance of the UAS from the GCS varies significantly across *different* traces, the variations are smaller *within* each trace. Hence, for Proteus, we pick the same $\overline{b}$ and α for each individual trace (corresponding to circular orbits at a certain distance from the GCS), though different parameters are used across different traces. For Proteus-Orient we also allow for different $\overline{b}$ and α for each orientation (*coming towards* and *going away*).

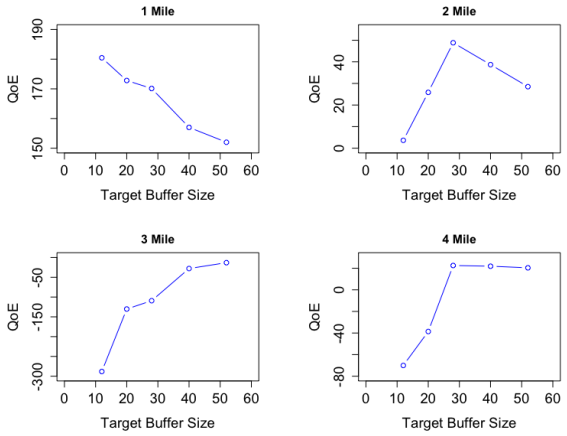

Figure 8. Target buffer size ($\overline{b}$) tuning for the circular orbits

To determine the best parameter choices, we simulate Proteus using different parameter settings, and pick the parameter resulting in the highest QoE for the entire video session. The test takes less than a minute. We choose the look-ahead optimization window to be $W = 5$ chunks, maximum bit rate $M = 4.3$, and we set $\lambda = 1$ and $\mu = 4.3$ so that 1 second of rebuffering penalty is equal to the maximum bitrate value (this method of parameter selection is consistent with previous work [16]–[18]). To illustrate our simulation, Fig. 8 shows how the achieved QoE varies based on the $\overline{b}$ parameter, while keeping α fixed at 3. The figure shows that in general larger distances require a larger target buffer size $\overline{b}$. This is due to higher dropouts and lower throughput as distance increases. We can see that low values of $\overline{b}$ are optimal for the 1 Mile distance because dropouts are not as disruptive at this distance. Proteus-Orient can further add performance by changing parameters based on orientation, as we show in §IV. As an example, one can see the optimal selection for Proteus for circ(3) is $\overline{b} = 52$ (with $\alpha = 3$), however, with Proteus-Orient, the optimal configuration becomes $\alpha = 3$ and $\overline{b} = 28$ for *going away*, and $\alpha = 5$ and $\overline{b} = 52$ for *coming towards*. This aligns with our flight measurements from §II, since the *coming towards* orientation induces more dropouts and lower throughput than the *going away* orientation, and a higher value of buffer occupancy would be necessary during times of degraded throughput.

We believe this approach is reasonable in practice because in aerial video coverage applications, it is common for a UAS to make multiple passes over a given area (e.g., on the same circular orbit). Thus, parameter choices may be informed using data gathered on an initial pass. For an initial pass, it is still possible to learn parameters from other previously flown trajectories and utilize them in current or future flights. To illustrate this, we consider the *Line* trace §II, and set its parameters based on those learnt from the *Circle* trace; we will present those results in the next section.

IV. Results

Next, we evaluate the benefits of our new designs, Proteus and Proteus-Orient. Unless otherwise mentioned, our results focus on the traces with omnidirectional antennas, MPC with the harmonic mean predictor, and a client buffer of 60 seconds. We also explore additional predictors and buffer sizes in a subsequent sensitivity analysis section.

A. Emulation methodology

We use Mahimahi [34] to emulate network throughput, replaying the various throughput traces collected from our flights. Since Proteus incorporates path-awareness and dynamically changes parameters over a given trace (e.g., based on distance and orientation), we modified the emulation setup to also include the distance and orientation information over time, along with the throughput.

The evaluations measure ABR performance using the "EnvivioDash3" video from the MPEG-DASH reference videos [35]. We chose this video because it has been extensively used in prior work [17], [18], [20], and since its length (193 seconds) is slightly larger than the time it takes to complete a full circular orbit flight pattern. The video is divided into 48 chunks, each of 4 second duration (with the last chunk slightly smaller). The video is encoded by H.264/MPEG-4 codec at bitrates of {300, 750, 1200, 1850, 2850, 4300} Kbps. We host the video on an Apache Server. Both the server and client software run on the same machine with Mahimahi performing the proper network emulation. The machine is a 64-bit Ubuntu 4-core Virtual Machine (VM) with 8 GB RAM. The ABR algorithm is implemented on a separate server process, and Dash.js configured to contact this process to determine the bitrates to fetch each chunk.

Our primary performance metric is the QoE metric from [16] presented earlier (Eq. (1), with i=1 and $j = N$, indicating that performance is measured over all video chunks). Although Proteus optimizes the modified metric in Eq. (2) in each look-ahead window, the original QoE metric is used to evaluate performance, as this captures the relevant metrics from the receiver's perspective. In addition, we present the constituent video delivery metrics (average bitrate, rebuffering ratio, and bitrate variations) to get a better sense of video performance.

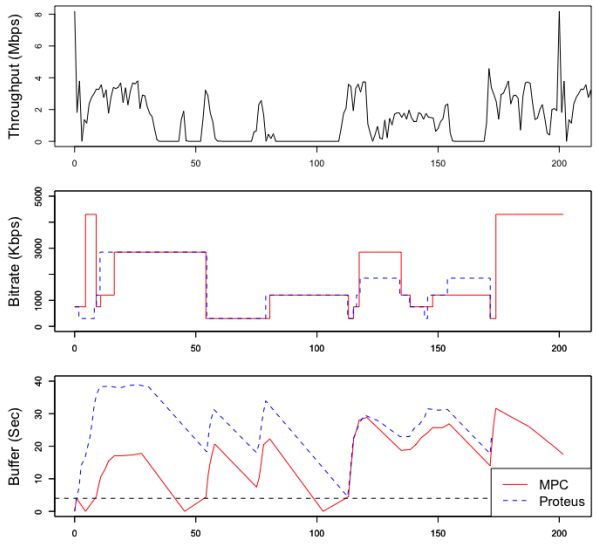

Figure 9. Proteus illustration for circ(4)

B. Benefits of Proteus

Fig. 9 illustrates MPC and Proteus for the circ(4) trace, showing the advantages of Proteus. The top figure shows throughput across time (notice the horizontal flat lines that represent dropouts), the second figure shows the video chunk bitrate, and the third figure shows the buffer (since each chunk contains 4 seconds of video, the buffer must have at least 4 seconds of video when a new chunk is selected for playback - otherwise there will be rebuffering. A dotted line is at the 4 second mark to easily identify rebuffering events). Due to its greedy nature, MPC can leave the buffer nearly empty (as discussed in §III), resulting in 3 rebuffering events (where the buffer drops below the dotted line). Each rebuffering event is several seconds, creating an undesirable and disruptive video streaming experience. We observe that Proteus is slightly more conservative than MPC, resulting in a larger buffer occupancy. This in turn allows Proteus to better handle large dropouts and avoid rebuffering while still achieving sufficient bitrate quality. Unlike MPC, the Proteus buffer does not drop below 4 seconds once the video streaming session begins.

Fig. 10 compares MPC with Proteus for the different circle traces. Each circle trace uses different insurance parameters, as described in §III. We noticed some variability across runs in our emulation tests. Hence, we test each trace and algorithm combination ten times and present a boxplot that shows the distribution of the QoE across the runs. Proteus outperforms MPC in all scenarios, except for circ(1). Here, the optimal parameter is $\alpha = 0$ (equivalent to no insurance), given that dropouts are less common. The results are dramatically improved with the circ(3) trace, which experienced the most dropouts. Further, notice that MPC exhibits significant variability because even minor processing delays across runs in the emulation can lead to slightly different bitrate selections, resulting in significantly different performance if a more aggressive choice were made just prior to a dropout. In contrast, Proteus is more robust to these variations because it can better overcome a poor bitrate selection choice with the larger buffer.

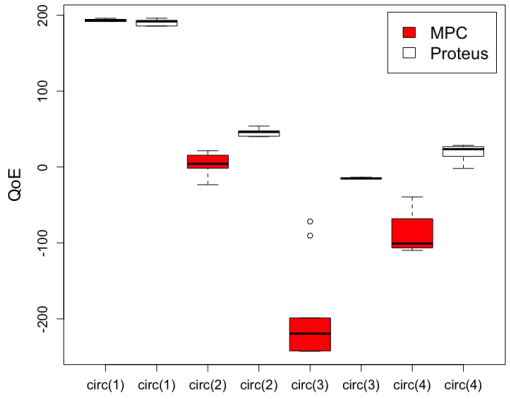

Figure 10. QoE benefits with Proteus

Fig. 11 presents a breakdown of the QoE metric into the constituent video delivery metrics. The top figure shows Proteus dramatically reduces rebuffering. The average rebuffering ratio is reduced from 10.06% to 0.38% for circ(2), from 23.97% to 8.16% for circ(3), and from 14.33% to 1.57% for circ(4). Further, these reductions are achieved without significant degradation in average bitrate (middle figure) and smoothness (lowest figure), since Proteus' selection of parameters are optimized for the dynamic UAS flight path.

C. Benefits of Proteus-Orient

Fig. 12 considers and shows the additional benefits if Proteus parameters were chosen based on orientation in addition to distance. For each circle trace, we determine two sets of parameters for each of the (*coming towards* and *going away*) orientations, and use the appropriate parameter based on the UAS orientation. There is a noticeable increase in QoE for the circ(2) and circ(3) traces. Further inspection showed that this was because Proteus-Orient achieved an increase in throughput by 13.34% and 14.38%, while reducing rebuffering ratio from 0.38% to 0.18% and from 8.16% to 5.82% for circ(2) and circ(3), respectively. Note that these benefits are in addition to significant gains already achieved by the Proteus scheme. Proteus-Orient does not provide additional benefits for circ(4) because the optimal values for $\bar{b}$ and α for circ(4) are the same for both orientations. We believe this is because each orientation for circ(4) experiences a similar average dropout

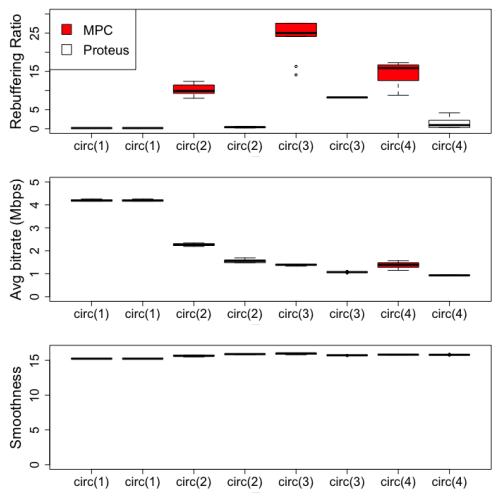

Figure 11. Breakdown of individual video delivery metrics

duration. Finally, we omit circ(1) because Proteus did not improve performance, due to less dropouts at this distance.

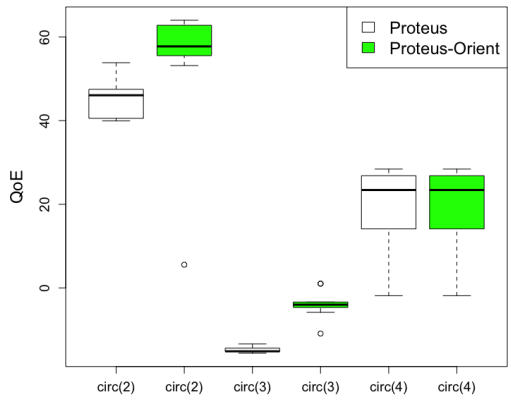

Figure 12. Benefits of considering orientation

D. Learning across traces

We next present results showing that parameters learned for Proteus in one environment can be effectively used in other similar environments. For this, we test Proteus for the *Line* trace, with parameters learned from multiple circular traces. Recall from §II that the *Line* trace is a straight line from a distance of 2.75 miles from the GCS to 0.75 miles. We use $\overline{b}$ and α from the circ(2) and circ(1) traces, dynamically monitoring the location of the UAS and adjusting the parameter values based on UAS position. We also compare this scheme to a "static optimal" scheme, which determines the best static $\overline{b}$ and α parameters for the *Line* trace. Fig. 13 shows that Proteus significantly improves QoE relative to MPC, and also achieves noticeable benefits over the static optimal. These results highlight the *benefits of dynamically adapting parameters with distance during the same flight path*, and also show the *feasibility of learning parameters from one trace and applying them to another*.

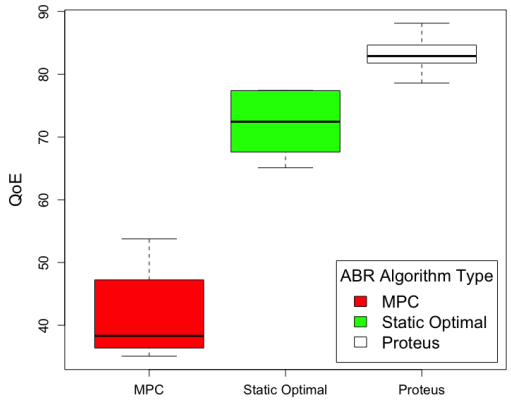

Figure 13. Proteus performance when parameters learned from one trace are applied in another trace and dynamically vary with distance

E. Sensitivity analysis

We evaluate if Proteus is still beneficial when: (i) a better predictor is used; (ii) a directional antenna is used; and (iii) the client buffer is smaller.

Predictor sensitivity: We have assumed network throughput prediction using the harmonic mean of previous samples. We evaluate whether Proteus benefits persist if better throughput prediction methods are used. We consider two such predictors: Hidden Markov Model (HMM) and an Oracle.

HMM: Recent work [36] has shown that network throughput follows different states (with each state corresponding to different throughput distributions). Based on this, a HMM predictor was developed, and was shown to perform better than harmonic mean for ABR algorithms. We tested Proteus and MPC with a HMM predictor and found that Proteus still outperforms MPC. The average QoE results were 201.4, 13.05, -206.5, -22.4 for MPC compared to 201.4, 47.84, -9.9, and 35.8 for Proteus for circ(1)-circ(4), respectively.

Oracle: Due to the greedy nature of MPC, it can perform poorly even with a perfect predictor, as discussed in §III. We next evaluate both MPC and Proteus with an Oracle that knows the exact throughput information for the duration of the look-ahead window.

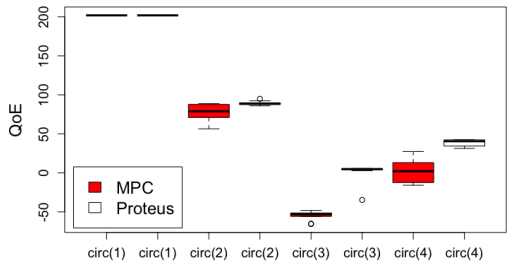

Figure 14. QoE benefits with Proteus (Oracle predictor)

Fig. 14 shows that Proteus still outperforms MPC in the QoE metric, even with perfect Oracle throughput prediction. Fig. 15 shows that Proteus significantly reduces the rebuffering ratio while only slightly reducing the average bitrate.

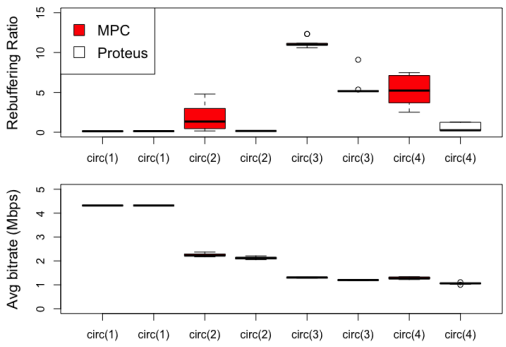

Figure 15. Oracle predictor performance measurements

Directional antenna testing: Directional antennas have higher gain than omnidirectional antennas, but at the cost of being larger and also having to be pointed at the other antenna. While this is not always practical, they are still used in some scenarios that permit the required logistics. While directional antennas have dropouts, they are typically shorter and an ABR algorithm can often recover from a myopic decision without rebuffering if a minimum video bitrate of 300 Kbps were used. However, if we consider higher quality bitrate requirements, then Proteus still shows significant benefits. For example, for the circ(4) directional trace and minimum bitrate of 1850 Kbps, Proteus reduced rebuffering and improved the average QoE by 34.6% (from 80.96 to 108.97).

Different client buffer sizes: We tested if Proteus benefits can be extended to smaller buffer sizes. To do so, we tested the traces again with buffer sizes ranging from 15 to 60 seconds. Proteus continued to show benefits. For example, with a buffer size of 30 seconds the average QoE results were 3.06, -194.45, and -59.27 for MPC compared to 15.28, -92.61, and -41.22 for Proteus for circ(2)-circ(4), respectively. These results show there is still significant improvement over MPC by using Proteus, even with smaller buffer sizes.

V. Related Work

UAS networking: A recent workshop paper [21] conducted a preliminary investigation of UAS for video transmission using ABR with content-based compression. Another paper [10] details optimal algorithms to improve sports surveillance with UAS. Several efforts are underway to use UAS to augment Internet connectivity in remote areas [11], [12], [37]–[39]. In such settings, there is freedom to position UAS to best serve the area [37]. There has been work with a focus on 802.11 fixed wing [40], [41] and multirotor [42]–[44] UAS networking, but at limited distances. Tactical Radios are tested with different transport protocols on the ground in [25] and in the air in [45], but without regard to specific applications. In contrast, our focus is on UAS for aerial video coverage, and enabling acceptable performance over extended distances, and when the location cannot be optimized for connectivity.

Video Streaming: The paper [46] uses measurement-based methods to adjust the video bitrate for video streaming. This work and ABR algorithms [17], [18], [20], [22], [36], [47] have been designed for traditional Internet environments and do not account for the challenges of dynamic long-range UAS flights. Recently, Elephanta [20] enabled edge users to provide feedback to the ABR algorithm via a QoE user perception interface and adaptation algorithm with flexible parameters, based on user preference. Oboe [18] has shown that it is feasible to improve the performance of state-of-the-art ABR algorithms, including MPC, by tuning their parameters (for MPC, the discount factor used to adjust throughput predictions) to network state. In contrast, we design Proteus to account for UAS flight path (distance and orientation), and handle dropouts by introducing a carefully selected terminal cost designed with parameter selection based on the UAS flight path. Pensieve [17] uses reinforcement learning to select bitrates. We are unable to evaluate Pensieve [17] in our settings given the lack of adequate training data, though our measurement and data collection efforts can facilitate the use of learning in the future. Finally, we expect that incorporating information about UAS flight path will benefit both learning approaches such as Pensieve [17], and buffer-based approaches [15], [22] (e.g., the minimum and target buffer thresholds in [22] can be configured in a manner informed by the UAS flight path).

VI. Conclusion

In this paper, we have taken key steps in enabling long-range UAS video streaming, with three contributions.

First, through analysis of data from real-world UAS flights, we have shown that extended dropouts make it challenging to simultaneously achieve sub-second video streaming latency and avoid significant loss of video content. However, our data shows the potential to achieve video streaming with modest delays (e.g., tens of seconds), and reveals how network throughput depends on flight path (both distance and orientation).

Second, we have presented Proteus, the first system for video streaming at long-range UAS distances. Proteus is based on a control-theoretic ABR algorithm approach and introduces a carefully constructed *terminal cost* into the receding-horizon optimization at each point in time. Proteus integrates knowledge of the flight path into its design, choosing terminal cost parameters as a function of both UAS distance and orientation.

Third, experiments on an emulation test-bed with real-world UAS flight traces show that Proteus significantly improves upon a representative ABR that has been shown to work well in Internet settings. For the circ(3) trace, which saw the most dropouts, the rebuffering ratio is reduced from 23.97% down to 8.16%, with a net QoE improvement from -198.84 to -14.72. Additionally, by taking advantage of UAS orientation, the rebuffering ratio is further reduced to 5.82%, and the QoE further increased to -3.83. The benefits hold across traces and distances, and even show the benefits of using Proteus with a perfect Oracle throughput predictor. Overall, the results show the feasibility of supporting demanding video applications in dynamic long-range UAS environments with Proteus.

VII. Acknowledgements

This work was partially supported by National Science Foundation (NSF) Award ICE-T 1836889, Cisco and NSF CAREER award 1653648. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF or Cisco.

References

[1] "FAA aerospace forecast fiscal years 2019-2039," Federal Aviation Administration, Tech. Rep., 2019.

[2] A. Hanscom and M. Bedford, "Unmanned aircraft system (uas) service demand 2015–2035, literature review & projections of future usage," *Res. Innov. Technol. Admin., US DOT, Washington, DC, USA*, 2013.

[3] O. Robert, "Small unmanned aircraft systems (SUAS) flight plan: 2016—2036," Washington, DC: US Air Force, Tech. Rep., 2016.

[4] Deloitte, "Managing the evolving skies. Unmanned aircraft system traffic management (UTM), the key enabler," 2019. [Online]. Available: https://www2.deloitte.com/global/en/pages/energy-and-resources/articles/managing-evolving-skies.html

[5] R. Dunkel, R. Saddler, and A. Doerry, "Use of unmanned sar and eo/ir sensor suites for monitoring wildfires," in *Radar Sensor Technology XXI*, vol. 10188. SPIE, 2017, p. 101881F.

[6] N. Dilshad, J. Hwang, J. Song, and N. Sung, "Applications and challenges in video surveillance via drone: A brief survey," in *ICTC*. IEEE, 2020, pp. 728–732.

[7] N. Search and R. Committee, "Unmanned aircraft system (uas) search and rescue addendum to the national search and rescue supplement to the international aeronautical and maritime search and rescue manual," Tech. Rep., 2016.

[8] S. Mitchell, "Unmanned aircraft flies first U.S. beyond-line-of-sight mission," 2019. [Online]. Available: https://news.uaf.edu/unmanned-aircraft-flies-first-u-s-beyond-line-of-sight-mission/

[9] "Drone delivery operations underway in 27 countries," 2019. [Online]. Available: https://www.unmannedairspace.info/latest-news-and-information/drone-delivery-operations-underway-in-26-countries

[10] X. Wang, A. Chowdhery, and M. Chiang, "Networked drone cameras for sports streaming," in *IEEE ICDCS*, 2017, pp. 308–318.

[11] "Google project loon," 2019. [Online]. Available: https://x.company/projects/loon/

[12] N. Lavars, "Verizon cell on wings," 2016. [Online]. Available: https://newatlas.com/verizon-drones-internet-trials/45818/

[13] "14 CFR: Aeronautics and Space," Code of Federal Regulations, 2019.

[14] A. Filippone, *Flight performance of fixed and rotary wing aircraft*. Elsevier, 2006.

[15] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *ACM SIGCOMM*, 2015, pp. 187–198.

[16] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *ACM SIGCOMM*, 2015, pp. 325–338.

[17] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with Pensieve," in *ACM SIGCOMM*, 2017, pp. 197–210.

[18] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: auto-tuning video abr algorithms to network conditions," in *ACM SIGCOMM*, 2018, pp. 44–58.

[19] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "ABR streaming of VBR-encoded videos: characterization, challenges, and solutions," in *ACM CoNext*, 2018, pp. 366–378.

[20] C. Qiao, J. Wang, and Y. Liu, "Beyond QoE: Diversity adaption in video streaming at the edge," in *IEEE ICDCS*, 2019, pp. 317–326.

[21] X. Wang, A. Chowdhery, and M. Chiang, "Skyeyes: adaptive video streaming from uavs," in *Proceedings of the 3rd Workshop on Hot Topics in Wireless*. ACM, 2016, pp. 2–6.

[22] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *IEEE INFOCOM*, 2016, pp. 1–9.

[23] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," vol. 36, no. 6. Elsevier, 2000, pp. 789–814.

[24] "Mpu specification sheet," 2016. [Online]. Available: http://www.persistentsystems.com/pdf/MPU4_SpecSheet.pdf

[25] M. Breedy, P. Budulas, A. Morelli, and N. Suri, "Transport protocols revisited," in *MILCOM 2015-2015 IEEE Military Communications Conference*. IEEE, 2015, pp. 1354–1360.

[26] M. Mozaffari, W. Saad, M. Bennis, Y.-H. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[27] iPerf, 2019. [Online]. Available: https://iperf.fr

[28] "Haigh-farr blade antennas," 2020. [Online]. Available: https://www.haigh-farr.com/docs/default-source/products-data-sheets/general_blade_booklet_rev1.pdf

[29] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 159–173.

[30] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 377–392.

[31] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Networks*, vol. 68, pp. 1–15, 2018.

[32] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *ACM Special Interest Group on Data Communication*, 2017, pp. 183–196.

[33] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM*, 2011, pp. 362–373.

[34] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for http," in *USENIX*, 2015, pp. 417–429.

[35] "Enviviodash3," 2016. [Online]. Available: https://dash.akamaized.net/envivio/EnvivioDash3

[36] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *ACM SIGCOMM*, 2016, pp. 272–285.

[37] A. Chakraborty, E. Chai, K. Sundaresan, A. Khojastepour, and S. Rangarajan, "Skyran: a self-organizing lte ran in the sky," in *ACM CoNext*, 2018, pp. 280–292.

[38] M. Moradi, K. Sundaresan, E. Chai, S. Rangarajan, and Z. M. Mao, "Skycore: Moving core to the edge for untethered and reliable UAV-based LTE networks," in *ACM MOBICOM*, 2018, pp. 35–49.

[39] K. Sundaresan, E. Chai, A. Chakraborty, and S. Rangarajan, "SkyLiTE: End-to-end design of low-altitude UAV networks for providing LTE connectivity," *arXiv preprint arXiv:1802.06042*, 2018.

[40] M. Asadpour, D. Giustiniano, and K. A. Hummel, "From ground to aerial communication: Dissecting wlan 802.11n for the drones," in *WiNTECH 2013*, 2013, pp. 25–32.

[41] M. Asadpour, D. Giustiniano, K. A. Hummel, and S. Heimlicher, "Characterizing 802.11n aerial communication," in *MobiHoc*, 2013, pp. 7–12.

[42] E. Yanmaz, R. Kuschnig, and C. Bettstetter, "Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility," in *INFOCOM 2013*. IEEE, 2013, pp. 120–124.

[43] S. Hayat, E. Yanmaz, and C. Bettstetter, "Experimental analysis of multipoint-to-point uav communications with ieee 802.11n and 802.11ac," in *PIMRC 2015*. IEEE, 2015, pp. 1991–1996.

[44] S. Hayat, C. Bettstetter, A. Fakhreddine, R. Muzaffar, and D. Emini, "An experimental evaluation of lte-a throughput for drones," in *DroNet 2019*, 2019, pp. 3–8.

[45] R. Shirey, S. Rao, and S. Sundaram, "Measuring fixed wing UAS networks at long range," in *6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications (DroNet)*, 2020.

[46] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "Qdash: a qoe-aware dash system," in *Proceedings of the 3rd Multimedia Systems Conference*, 2012, pp. 11–22.

[47] L. P. Koh and S. A. Wich, "Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation," *Tropical Conservation Science*, vol. 5, no. 2, pp. 121–132, 2012.