DATA-DRIVEN AND CONTROL THEORETIC APPROACHES FOR REAL-TIME SENSOR DATA TRANSMISSION OVER UNMANNED AERIAL SYSTEMS NETWORKS

by

Russell Shirey

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Sanjay Rao, Co-Chair

School of Electrical and Computer Engineering

Dr. Shreyas Sundaram, Co-Chair

School of Electrical and Computer Engineering

Dr. Vijay Raghunathan

School of Electrical and Computer Engineering

Dr. Daniel DeLaurentis

School of Aeronautics and Astronautics

Approved by:

Dr. Dimitrios Peroulis

To my family.

ACKNOWLEDGMENTS

I would like to sincerely thank my advisors, Dr. Sanjay Rao and Dr. Shreyas Sundaram, who have both been great mentors and colleagues during my time at Purdue University. I am indebted to both of them for their helpful wisdom, guidance, and support along this PhD journey. Amidst challenges, the Covid pandemic, and the ups and downs of this journey, they continued to meet on a regular basis and were always supportive and available whenever needed. I have learned more than I would have ever imagined from studying with them as my advisors, and am truly thankful for the opportunity to work with them both.

I am also very thankful for my committee members, Dr. Vijay Raghunathan and Dr. Daniel DeLaurentis, and grateful for the exposure to new ideas and valuable feedback they have provided me on my research. I look forward to continuing a professional and productive collaborative relationship with you all, and Purdue University, throughout my career.

I am indebted to both the Air Force and Purdue Military Research Institute (PMRI) for allowing me the opportunity to pursue doctoral studies at Purdue University. I look forward to using my newly gained knowledge in service to our great nation.

I am fortunate to have worked with such talented fellow student colleagues in our research groups: the Internet Systems Lab and the Center for Innovation in Control, Optimization, and Networks. It has been a pleasure getting to know and work with you all over the past few years, and I wish you the best.

Last but not least, I would like to thank all of those at Purdue who have played a part in this journey. I am truly grateful for my experience at this fine university.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Unmanned Aerial Systems (UAS) are often used to collect and transmit sensor data (e.g., video, radar images) to the ground. While much research related to data transmission in UAS settings has focused on short distances, there is growing interest in operating UAS beyond Visual Line of Sight (VLOS), a relatively unexplored research area. In this thesis, we make three contributions. We present one of the first characterization studies of UAS network performance when operating at distances exceeding VLOS. Our results confirm challenges owing to wireless network variability but also point to opportunities to exploit the correlation of network performance with flight path (distance and orientation). Second, motivated by our observations, we design Proteus, the first system for video streaming targeted at long-range UAS settings. Proteus is distinguished from existing algorithms developed for traditional Internet settings by explicitly accounting for dropouts, and leveraging flight path information. Through flight emulation experiments, we show Proteus reduces rebuffering from 14.33% to 1.57% at long-range distances, while significantly improving composite video delivery metrics. Third, we design Chimera, which uses the flight path to optimize heterogenous sensor data transmission. Chimera is based on an optimal control framework, performing online optimization to yield a feedback control policy that makes transmission decisions. Through emulation and simulation experiments, Chimera reduces penalties related to dropped radar images by 72.4%-100%, compared to an algorithm agnostic of flight path, and achieves an average bitrate of 90.5%, compared to an optimal scheme knowing future throughput, with only minimal increase in radar images dropped.

1. 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. As a result, UAS usage has grown at incredible rates and they are often used to provide sensing and data-gathering in a variety of scenarios, due to their ability to go to areas where humans cannot, and gain vantage points and coverage with sensors that are not possible from the ground [1]–[6]. The types and capabilities of sensors, and options for mounting on UAS, have increased to provide many options to gain insight from an aerial viewpoint [4], [7]–[9]. There is growing interest in operating UAS at ranges beyond Visual Line of Sight (VLOS) since it increases the UAS mission opportunities [1], [10], [11]. UAS are broadly used in many domains including military [7], [12], disaster response [2], [8], [13], search and rescue [14], law enforcement [2], agriculture [2], railroad and pipeline inspection [2], [10], package delivery [2], [11], conservation management [2], and other domains [2], [11], [15].

1.1 UAS wireless data transfer challenges and opportunities

Transmitting sensor data from an UAS to a Ground Control Station (GCS) in real-time presents **challenges** due to flight dynamics and bandwidth limitations of the wireless network [4], [5], [16]–[19]. However, there are also **opportunities** for improvement, since the wireless network performance of the UAS depends on the flight path (which we explore and present observations in our analysis with real-world UAS flight data that we collected in §2). In the rest of this thesis, we use the term **UAS network** to refer to data transmission from an UAS to a ground node (e.g., GCS). Swarms of UAS nodes are out of scope for this dissertation.

Despite being a different and significantly more challenging environment than traditional computer networks (e.g., Internet), there currently exists limited understanding of long-range UAS networking (and its implications for applications). Thus, it is both important and necessary to collect real-world flight data to properly analyze and understand these networks, and how they operate over long periods of time and at ranges beyond VLOS, in order to design relevant and improved sensor data transmission algorithms and systems.

1.1.1 Dynamic flight network challenges

There are multiple different types of wireless networking options for UAS flight - each with different pros and cons (as we discuss in §2.2). Since the UAS must travel to locations determined by the application requirements (e.g., disaster relief in a specific area, or providing mapping and oversight of an area to guide search and rescue missions), it is desirable to have long-range options, and there is relatively limited freedom in placing UAS to optimize for connectivity. Today, civilian use of UAS in the United States (US) is typically restricted to limited distances that require VLOS [20] (typically, 1 Km). However, there is much interest in going to larger distances and regulations are beginning to support this worldwide [1], [10], [11]. Our work focuses on long-range applications beyond VLOS, a relatively unexplored research area due to the challenges of collecting such data, but an area that presents advantages for UAS mission scenarios. While this long-range focus increases the benefits of flight, it also makes collection of data more challenging. There is limited prior research work to leverage due to the challenges of data collection at extended distances. Further, extending the distances of UAS networks results in degradation of wireless network performance. Additional challenges include variable performance based on the type of UAS (fixed wing or multirotor) [21], type of antenna (omnidirectional or directional), and the flight path of the UAS. We discuss all of these critical aspects in §2.

1.1.2 UAS sensor challenges

A wide variety of different sensors are used for UAS missions. Examples include video (with different lenses and modes, such as infrared), Synthetic Aperture Radar (SAR), and Light Detection and Ranging (LiDAR) [7], [22], [23]. Many UAS settings, such as security surveillance [7], [24], search and rescue missions [14], [22] that are aided with wide-range SAR imagery accompanied by live video, and environmental monitoring [23], [25], involve both video and other sensors such as radar imaging (e.g., SAR) [7], [22], [23]. Video provides color imaging of small visible areas, while SAR imagery provides a wide-area all-weather capability that penetrates fog, smoke, and atmospheric obstructions [23], [26]–[28] (e.g., the ability to penetrate through smoke is critical in fire monitoring scenarios [8], [9]). UAS networks are often not able to keep up with demand in transferring important sensor data in flight to the ground (especially at long-range distances and/or if multiple sensors are being used in parallel). Sensor data can be gathered on the UAS and offloaded once on the ground and connected to traditional Internet. However, it is usually more valuable and efficient to offload the data during the flight to guide the flight mission and provide insight into real-world events, disasters, research, and more. While data and quality of these different types of sensors vary, the common desire is to have more data, faster access to the data, and higher quality data (which is larger in terms of size). Additionally, being able to access the data in a reliable manner is desirable for most of the data types, although some losses can be tolerated, depending on the sensors and circumstances.

1.1.3 UAS flight opportunities for improvement

While dynamic UAS flight sensor data communication networks present challenges, especially at long-range distances, there are also significant opportunities that can be leveraged in order to improve performance. We discuss opportunities that can be used to take advantage of the dynamic networking performance based on UAS flight path in §2. We explore different UAS sensor data networking applications for transmitting sensor data to the ground, and design new improvements to these types of applications, based on our measurements and analysis of real-world UAS flight tests, collected at multiple locations across the United States (US). Specifically, we explore the popular UAS application of recording video from the UAS and streaming this video to the ground, at long-range, in §3, designing a new system to optimize for the dynamic nature of UAS flight. Further, we expand into multiple heterogenous sensor applications and develop a long-range multi-sensor SAR and live-video streaming application in §4.

1.2 Current state of the art

We next provide an overview of the current state of the art and prior work in this area, given the aforementioned growth of UAS and aerial sensor surveillance systems. There have been recent measurement studies for UAS wireless communications [29]–[35], but these have all either been based on theory, without real-world measurements, and/or at limited distances (within 0.25 miles), or with LTE that requires the UAS to be tethered to infrastructure, limiting where the UAS can fly and communicate. Further, there has been much work in video streaming, but most of this work is in the context of the traditional Internet [36]–[40]. Some video streaming studies have been performed with UAS [15], [17], [41], but at limited distances and functionality. The challenges in this thesis are focused on long-range UAS mission sets (e.g., exceeding VLOS) that optimize communication with different types of sensors, with the ability to adapt to the UAS flight path in order to optimize data transmission, motivated by real-world flight datasets.

1.3 Contributions

This work takes key steps to extend UAS sensor data transmission applications to distances exceeding VLOS. We present three new contributions, described below:

1.3.1 New measurements and characterization of long-range UAS networks

First, we present one of the first characterization studies of UAS networking at long-range distances beyond VLOS [42], a relatively unexplored research area due to civilian flight regulations and the difficulty of flying at such range. Our tests are conducted through special approval and coordination, consisting of UAS flights tests over multiple days with both fixed wing and multirotor UAS at locations in Florida (FL) and California (CA). While there have been network studies with UAS, most have been conducted with multirotor UAS and previous work focuses on experimentation at shorter distances (less than 0.25 miles), with a mixture of 802.11 [29]–[31], [35], [43] and LTE [33], [34], [44]–[46] networking technology focus.

Our measurements focus on **long-range distances** with both **fixed wing and multirotor UAS**. Additionally, our experiments are conducted using **Tactical Radios** [42], [47], which do not rely on pre-existing infrastructure, thus enabling more flexibility in their applications. We are motivated to understand networking capabilities and limitations with UAS operating at long-range distances, and with different omnidirectional and directional antenna configurations.

Key findings: Our measurements show how network throughput, and periods of dropout (where no data goes through), varies with flight path. Interestingly, the data indicates the **orientation of the plane relative to the ground node also significantly impacts network performance** with both types of UAS. We also closely explore long-term flight plans, where we can exploit the UAS flight path to improve data transmission and mission performance in critical ranges beyond VLOS. In these scenarios, we analyze situations at the edge of remote connectivity, where dropouts are common. Understanding the performance under these different scenarios will extend the usable distance of UAS, and greatly increase their applicability.

1.3.2 Proteus, the first video streaming system designed for long-range UAS flight settings

Second, motivated by the observations above, we present Proteus, a system for video streaming in long-range UAS settings [48]. Proteus leverages Adaptive Bit Rate (ABR) algorithms [36], [37], [40], [49]–[51], 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 [15], [41]. To our knowledge, Proteus is the first system for video streaming that tackles long-range UAS settings, and issues unique to UAS flight.

Key findings: Proteus is based on a control-theoretic approach, motivated by the success of such approaches for traditional Internet video streaming [39], [50], [51]. Unfortunately, we illustrate that a direct application of a representative and widely studied ABR algorithm based on a control theoretic approach [50], 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 the introduction of 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 [52]; 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 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).

Results: 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 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.

1.3.3 Chimera: exploiting the UAS flight path to optimize simultaneous sensor data transmission

Third, motivated by our measurements, we developed Chimera, a system that taps into the opportunity to exploit the UAS flight path to improve sensor data transmission. Chimera optimizes transmission of heterogeneous sensor data over variable UAS network

[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 [50], following convention in the video streaming community.

environments at long-range and extended periods of flight. Chimera is based on an optimal control framework, performing online optimization in order to yield a feedback control policy that makes transmission decisions for the two different sensor data streams: (i) video, and (ii) Synthetic Aperture Radar (SAR) images. While we focus on these streams for concreteness, Chimera can be generalized to more diverse data streams as well.

Key findings: Chimera develops and validates robust UAS network throughput prediction and error models through a detailed and data-driven analysis of our UAS flight test data. A key consideration for providing useful models that can be used in working systems is developing a pragmatic model whose parameters can be learnt online using information from initial stages of the flight. We explore and analyze models using different combinations of flight and throughput parameters. However, our analysis indicates that simple regression models based on the distance and UAS orientation relative to the Ground Control Station (GCS) are effective in prediction, even into the future. We integrate our models into Chimera, which learns both the dependence of throughput on flight path, and also an error model pertaining to throughput prediction errors. Chimera's approach is viable since flight paths are typically determined in advance. Further, its optimal control framework uses a continual planning model, which allows it to adapt to flight path changes by learning and improving throughput and error models over time.

Results: With a combination of emulation and simulation experiments using real-world flight traces, we show Chimera's effectiveness. Specifically, Chimera reduces penalties related to dropped radar images by 72.4%-100% compared to an algorithm agnostic to flight path information, and achieves an average bitrate of 90.5% compared to an optimal scheme that knows the exact future throughput, with only a minimal increase in radar images dropped.

1.4 Thesis organization

This thesis is organized as follows. Chapter 2 presents data and analysis from real-world UAS flight tests. These tests are conducted with both fixed wing and multirotor UAS, and at ranges exceeding traditional civilian regulations, through special approval. This work provides unique insights into performance characteristics of UAS networks at long-range. Our recorded network measurements provide relevant data and a basis for research to improve dynamic UAS sensor data transmission algorithms. Chapter 3 presents Proteus, the first system for optimization of UAS video streaming at long-range distances. Proteus utilizes control theory, modern video streaming algorithms, and knowledge of the UAS flight path to uniquely tailor parameters to optimize a new and improved design architecture. Chapter 4 presents Chimera, a system that taps into this opportunity while transmitting heterogeneous data streams over UAS networks. Chimera presents detailed analysis into throughput prediction and errors models, and learns a model online that relates UAS network throughput to the flight path. Chimera then combines the model with a control framework that optimizes simultaneous transmissions from different sensors, based on long-range throughput prediction. Finally, we conclude and propose future work into the areas of UAS swarms, satellite communication (SATCOM) with UAS, experimentation with our algorithms in 5G wireless settings, and enhancements using Artificial Intelligence (AI).

2. MEASURING FIXED WING UAS NETWORKS AT LONG RANGE

2.1 Introduction

Recent technological advances have increased Unmanned Aerial Systems (UAS) usage at high rates and show no signs of stopping [1]. Many UAS applications involve sensor data collection and transmission of sensor data to an interested party on the ground, given that UAS can fly and go to places that humans cannot due to lack of physical access or as a safety precaution (e.g., areas impacted by disasters, law enforcement, etc.). The sensor data often has reliability requirements. Examples of such sensor data can include video with reliability requirements, Synthetic Aperture Radar (SAR), and Light Detection and Ranging (LiDAR) [7], [22], [23]. These sensors are often combined, thus putting significant demand on the network. Reliable networking is required to ensure important information is not missed due to dropped packets.

As discussed in §1.1, UAS must often travel to locations determined by the application requirements (e.g., providing mapping and oversight of an area to guide search and rescue missions), resulting in relatively limited freedom in placing UAS to optimize for connectivity. Recall from §1 that civilian use of UAS in the US is typically restricted to limited distances that require Visual Line of Sight (VLOS) [20] (typically, 1 Km). However, there is much interest in going to larger distances and regulations are beginning to support this worldwide [1], [10], [11].

Contributions: In this chapter, we conduct a detailed measurement study of UAS network communication to a Ground Control Station (GCS). The study is done at two different locations across the US, each over multi-day periods, and using both fixed wing and multirotor UAS at distances exceeding traditional civilian regulations, through special approval. While there have been network studies with UAS, most have been conducted with multirotor UAS and at limited distances (less than 0.25 miles). In addition to shorter distances, previous work focuses on experimentation with a mixture of 802.11 [29]–[31], [35], [43] and LTE [33], [34], [44]–[46] networking focus. Our measurements focus on **long range distances** with both **fixed wing and multirotor UAS**. Additionally, our experiments are conducted using

Tactical Radios [42], [47], which do not rely on pre-existing infrastructure. We are motivated to understand networking capabilities and limitations with UAS operating at long range distances, and with different omnidirectional and directional antenna configurations. Our measurements show how network throughput, and periods of dropout (where no data goes through), varies with flight path. Interestingly, the data indicates the **orientation of the UAS relative to the ground node also significantly impacts network performance** for both types of UAS. We explore long-range flight paths and carefully examine the dynamics of UAS flight and its effect on network throughput. We also pay close attention to situations on the edge of remote connectivity, where dropouts are common. Understanding the performance under these different scenarios will extend the usable distance of UAS, and greatly increase their mission capabilities.

2.2 Background

Fixed wing vs. multirotor UAS: There are two broad kinds of UAS - fixed wing and multirotor systems [21]. 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. Both systems are used in practice. Multirotor UAS can provide greater flexibility, with vertical take-off and the ability to change directions very quickly. Fixed wing systems benefit aerial coverage applications from typically being faster and having longer endurance than their multirotor counterparts. However, both systems can be challenging to characterize from a Radio Frequency (RF) standpoint. Fixed wing UAS are not symmetrical in shape (leading to potentially varying wireless networking performance based on orientation), and typically must remain in motion in order to stay aloft. Multirotor UAS are often symmetrical, but can tilt (vertically, or in attitude) as they fly, potentially causing obstructions from an RF standpoint.

Tactical Radios vs LTE/WiFi: Many aerial surveillance coverage applications cannot rely on pre-existing infrastructure, and hence require the entire wireless infrastructure on the UAS and the ground node. We refer to **Tactical Radios** [42], [47] as radios that integrate the full infrastructure needed for communication into the radio, allowing them to be interchangeable and operate in an ad-hoc manner. These radios utilize Mobile Ad-Hoc Networking (MANET) with multiple nodes, and can also operate in a point-to-point mode, with two dedicated radios for network traffic. This is relevant for UAS aerial coverage, with a radio on a UAS transmitting sensor data to another on the ground.

Tactical Radios have been used in the military for years and have since been adopted by Government and commercial entities, due to their capabilities. They are now commonly used to achieve goals in areas such as disaster relief [53], fighting wildfires [54], law enforcement [55], and crowd management and surveillance [56]. They have also been used to provide coverage of sporting events, such as the Super Bowl [57]. Due to their flexibility and applicability for use in long-distance UAS applications, we focus on Tactical Radios in our research.

While there is recent interest in mounting LTE base stations on UAS [34], the technology is still under development. Furthermore, LTE has different infrastructure requirements based on whether the communication node is the master or slave. It is advantageous to have consistent networking equipment that is interchangeable and has both the master and slave functionality. This ensures that less backup equipment is needed, since it is not specialized, lowering the logistics footprint. Finally, the Tactical Radios we consider have much longer range than WiFi, presenting more options for aerial coverage at extended distances.

Types of antennas: UAS often operate with omnidirectional antennas, sometimes just referred to as **omni**, due to their constant movement. Omnidirectional antennas on the ground are the best for application flexibility, since they do not have to point at the UAS (and thus do not need a tracker). This enables the omnidirectional antenna to be mounted almost anywhere, even on a person, and moved around at ease. However, they have lower gain, resulting in less throughput than directional antennas. Directional antennas are larger and require a tracker to direct the antenna to the UAS, resulting in more equipment and setup time. There are trade-offs in either case and both are widely used in practice, depending on the application. Fig. 2.1 shows a comparison of a directional and omnidirectional antenna that we used in our testing.

2.3 Measurement methodology

We seek to understand network performance in UAS settings, with the intention of enabling aerial surveillance with sensors, transmitting the sensor data to the ground over distances that stretch the limits of wireless connectivity, with both fixed wing and multirotor UAS. We characterize how network performance varies with distance, UAS orientation, and antenna type. We present our measurement methodology, and discuss the data collected.

Regulatory approval: UAS flight in the US is governed by the FAA [20]. Under civilian regulations, UAS are typically restricted to an altitude of 400 ft, and to distances that require visual line of sight for the duration of a flight (typically, 1 km). Given flight regulatory restrictions in the US, we collaborated with the Air Force Research Laboratory (AFRL) to accomplish relevant fixed wing UAS flight testing at distances exceeding current FAA limits (recall we expect the FAA limits to increase over time (§2.1)). We also collaborated with a flight business and utilized strategic placement of our GCS (described below) to extend the wireless network range of the multirotor UAS flights.

Flight terminology: We introduce the flight terms **Distance, Slant Range, and Altitude**, and show them in Fig. 2.2. The plane orientation of **coming towards** (to the GCS) and **going away** (from the GCS) are also shown.

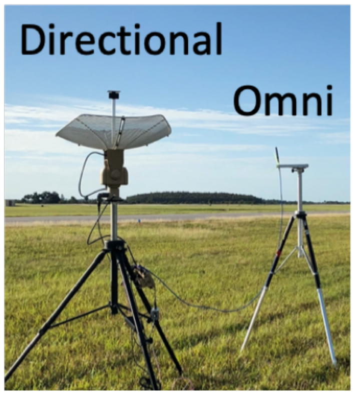


Figure 2.1. Antennas

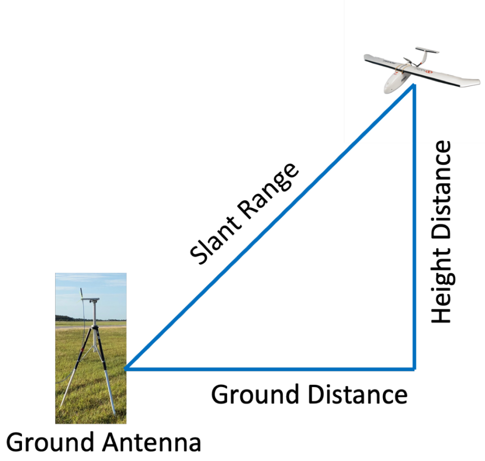


Figure 2.2. Orientation

Hardware setup: Both UAS that we tested with had areas for storage. The fixed wing UAS contained an air cooled payload bay on the bottom of the plane and the multirotor had bracket mounting points. We stored a Raspberry Pi connected to our radios in order to transmit and receive network data to and from the ground. Our Raspberry Pi also enabled storage of data on the UAS (using a 32 GB SD Card). On the ground, we used a laptop connected to the appropriate radio to communicate with the UAS.

Flight test locations: We collected UAS flight test wireless network data at distances exceeding VLOS over multiple day periods in Florida (FL) and California (CA), described below:

2.3.1 Florida dataset

Aircraft selection: We flew a Martin UAV Bat-4, a representative fixed wing UAS in terms of size, weight, and speed, and appropriate for aerial surveillance activities. The Bat-4 flies 40-70 knots, depending on weight and wind conditions. The pilot controlled the airplane from the Ground Control Station (GCS) using a separate Command and Control (C2) link at a lower frequency that did not interfere with our data link.

Flight patterns: Circular orbits around a point are useful for recurring coverage of an area, and are common across many surveillance applications. For this reason, we mostly flew circle patterns to the east of the ground station, as shown and described in Fig. 2.3. We refer to the data collected from the circle orbits as **circ(k)**, with $k \in \{1, 2, \ldots, 7\}$, depending on the distance of the center of the circle from the GCS. We flew at 1500 ft cruising altitude Above Ground Level (AGL) and typically within 50-60 kts airspeed (taking about 160-180 seconds per circle orbit). With clear vision from the GCS to the UAS, the slant range determines the absolute distance between the radios. This range affects the signal strength, and subsequent performance, of the network. Since slant range incorporates both ground distance and altitude into the calculation, we were able to test a wide variety of slant ranges while keeping altitude constant at a level that allowed for unobstructed point-to-point communication between the ground radio and the UAS.

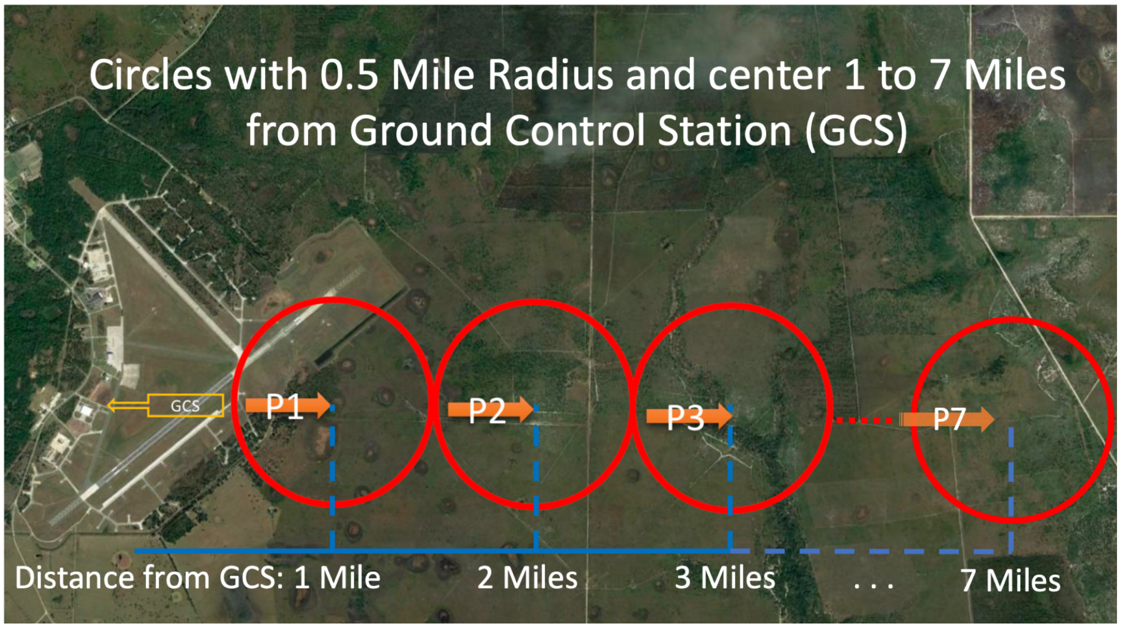

Circles with 0.5 Mile Radius and center 1 to 7 Miles from Ground Control Station (GCS)
GCS
P1
P2
P3
P7
Distance from GCS: 1 Mile
2 Miles
3 Miles
. . .
7 Miles

Figure 2.3. FL flight plan and patterns

Radio Selection: We used two Persistent Systems MPU4 Tactical Radios [58] tuned to S-Band frequency for testing. These radios provide seamless long-range layer 2 connectivity and support Internet Protocol (IP) traffic. We chose this configuration since the UAS sensor applications we consider require two radios in a point-to-point configuration: one on the UAS and the other on the ground.

Antenna Selection: We tested with an omnidirectional antenna on the plane and with both directional and omnidirectional antennas on the ground (see Fig. 2.1), since both are widely used in practice. We used a large directional antenna with 27 dBi gain on the ground (L-Com (HG2427)). This antenna aperture is 47.2 inches (in) x 35.43 in, offering significant gain but is very large and not practical for many applications. We used power ranging from 2W to 63mW, effectively lowering the gain to 12 dBi at 63mW, to be comparable to a smaller directional antenna, appropriate for more applications. We focus on this configuration throughout most of this chapter, as the data collected with this configuration is more relevant to the edge of connectivity, given our flight distances. We use the term "directional antenna" to represent this configuration. A 12 dBi directional antenna is still larger than the omnidirectional and also requires a tracker to accurately point to the UAS. We used a small Haigh Farr 6130-4 omnidirectional blade antenna on the UAS.

2.3.2 California dataset

Plane Selection: We flew and collected this dataset in CA in partnership with a local flight business. We used one of their industry-grade multirotor UAS, a good fit for the missions we have described.

Flight patterns:

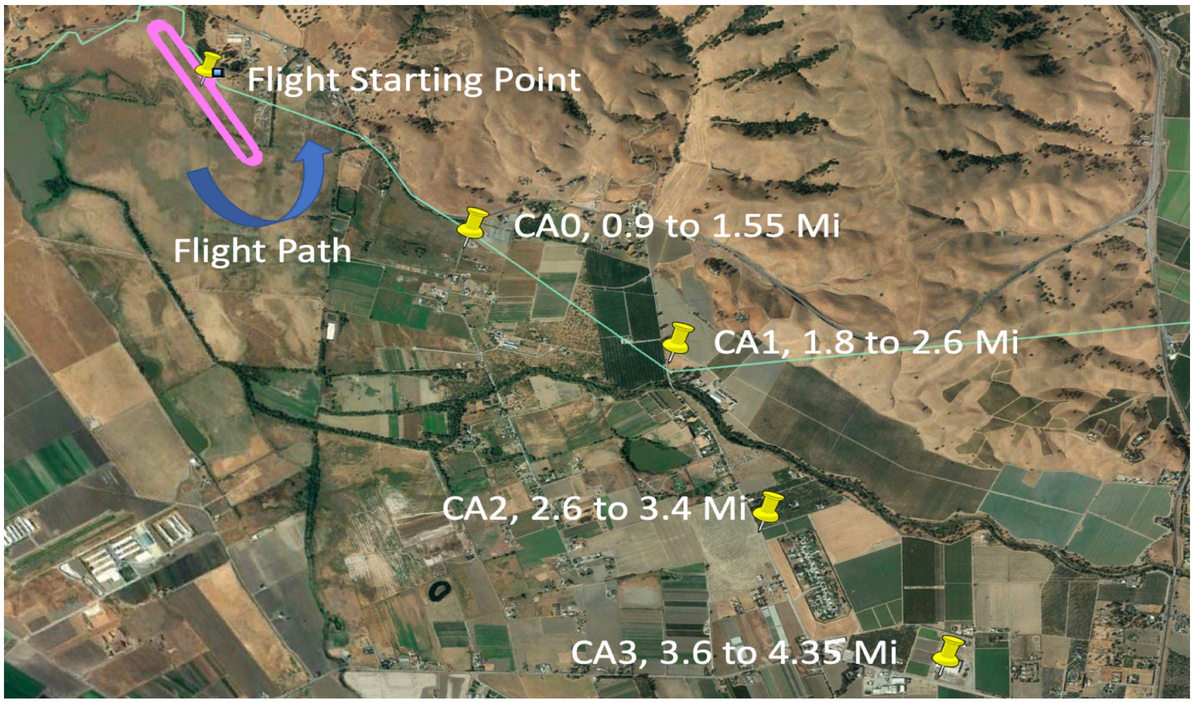


Figure 2.4. CA flight plan and patterns

Fig. 2.4 shows the flight setup for our CA dataset collection. We had permission to fly in a range of roughly 0.8 miles (due to private land). To extend the collection range, we moved our GCS (co-located with our car), flying a full flight and collecting several loops of data (relevant for aerial surveillance) at each GCS location. The entire dataset was collected at four different GCS collection locations at distances ranging from 0.9 to 4.35 miles. The flight used a multirotor UAS [21] flying in an oval racetrack orbit (similar to a circle orbit, but extended length-wise) at speeds of 15-25 knots, slower than the fixed wing alternative that we collected data with in FL.

Radio Selection: We used two Trellisware Shadow Tactical Radios [59] tuned to S-Band frequency for testing. Like the MPU4, these radios also provide seamless long-range layer 2 connectivity and support Internet Protocol (IP) traffic. We likewise chose this configuration since the UAS sensor applications we consider require two radios in a point-to-point configuration: one on the UAS and the other on the ground.

Antenna Selection: We tested with omnidirectional antennas on the plane and on the ground, for flexibility due to the movement of our GCS (co-located with our car). We used 2W power for maximum range and connectivity, and used Trelliware antennas on the UAS and for our GCS.

2.3.3 Data collection

We scheduled flights on multiple different days to accomplish our goals for each test. We flew all flight patterns for a given configuration in a single day to keep consistent results. We collected second-by-second throughput, latency, SNR, and location data with synchronized clocks. Our throughput was collected using iPerf [60]. The FL dataset contains TCP throughput information, while the CA datasets are based on mostly UDP measurements (with some TCP measurements included as well). We hosted the iPerf server on the Raspberry Pi in the UAS and ran the iPerf client on the ground laptop. We recorded latency via pings from the ground laptop to the Raspberry Pi. We recorded SNR and GPS information directly from the UAS radios and GPS antenna on the UAS, respectively.

2.4 Data analysis

We first analyze each dataset separately for clarity, and then detail our takeaways from the analysis of both.

2.4.1 FL dataset analysis

Our FL testing culminated in 6,245 seconds of throughput test time and 37 individual traces. The average throughput of each circular orbit test (0.5 Mile radius and center of circle 1 to 4 miles from GCS) is shown in Table 2.1 (the distances beyond 4 miles were only collected with the directional 500mW and 2W configurations).

Table 2.1. Circle orbit TCP average throughput (Mbps)

Orbits With Center X Miles From GCS				
Config	**1**	**2**	**3**	**4**
Omni 2W	5.03	2.31	0.97	1.32
Dir 2W	15.40	13.20	12.00	10.80
Dir 500mW	14.60	13.00	11.70	10.10
Dir 125mW	14.60	11.50	9.08	6.92
Dir 63mW	10.80	6.90	4.92	3.26

As expected, we see that throughput decreases as distance increases (except for the anomaly of the omnidirectional 4 mile test, which will be explained later). Another observation is that the directional configurations all performed significantly better than the omnidirectional. This is expected because even the lowest gain directional configuration (63mW) is equivalent to a 12 dBi antenna operating at 2W, much higher than omnidirectional. Table 2.1 can help guide applications for UAS operations with different configurations. In this chapter, we focus on the 63mW directional and 2W omnidirectional configurations, as they present edge cases of network performance with challenges to consider when designing UAS applications.

Network performance over the flight path: Fig. 2.5 and Fig. 2.6 show a time series of the slant range of the aircraft, as well as the network metrics (throughput, latency, and SNR) for the circ(4) trace with omnidirectional and directional antenna configurations, respectively. The figures allows us to observe network performance, and how it varies with slant range. Dropouts are more prevalent at this distance, especially during the **coming towards** duration (from about 50 until 140 seconds for each circ(4) trace). Dropouts can be seen in Fig. 2.5 and Fig. 2.6 as sections where the throughput is 0 (top-left plot). Additionally, dropouts cause the lines to disappear in the other three plots. While the

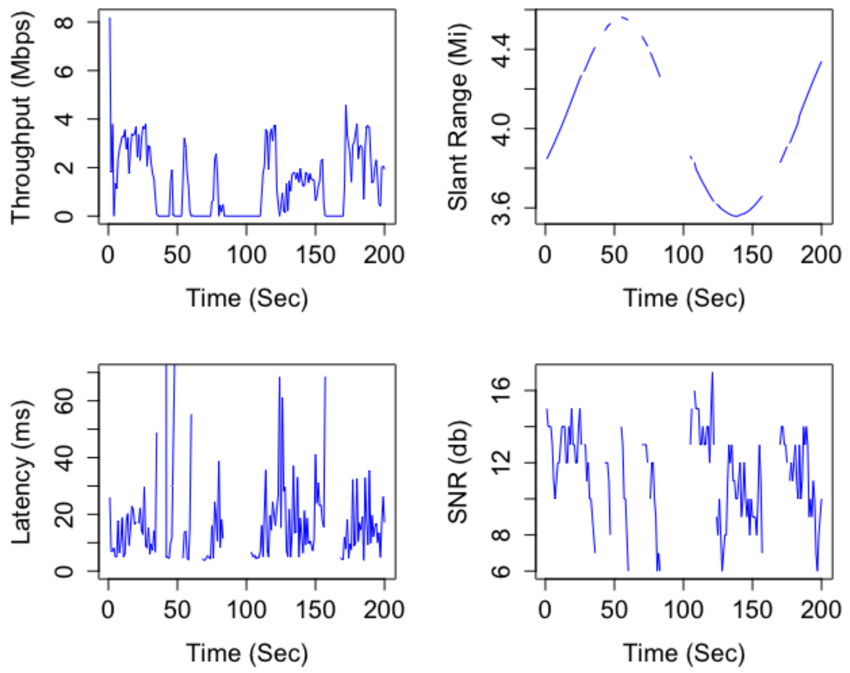


Figure 2.5. Omni circ(4) metrics

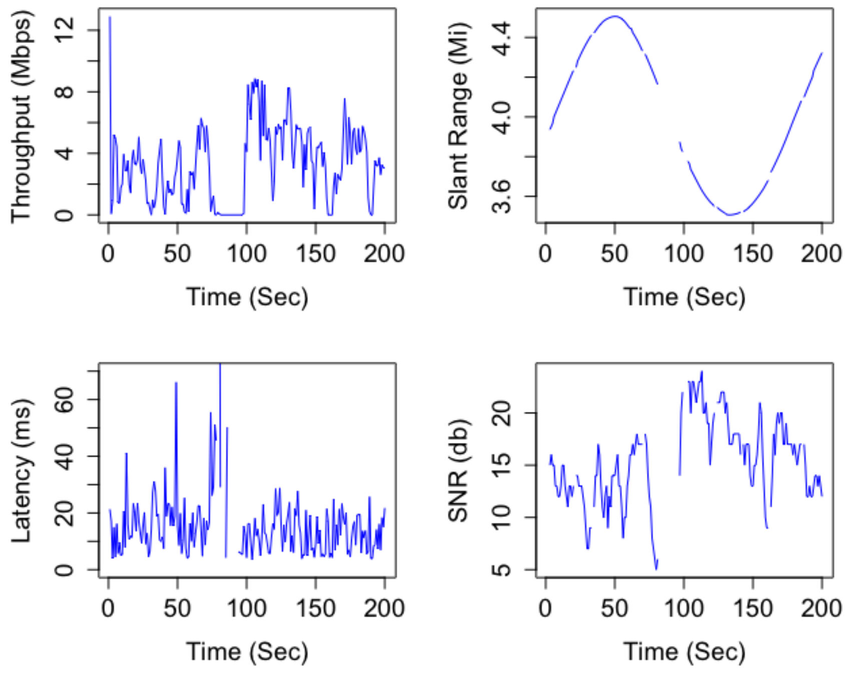


Figure 2.6. Directional circ(4) metrics

latency does not have a distinct pattern, **we observe several dropouts in all test cases.** Of note, we observe an increase in dropouts while the UAS is in the **coming towards** orientation (especially between 80-100 seconds) of the traces; this qualitatively suggests that the orientation of the UAS will have an impact on network performance. We will see that this impact is indeed present, and will quantify the differences in the subsequent subsections. The other omnidirectional traces and corresponding directional traces were qualitatively similar, with less dropouts as the range of distance decreased.

Fig. 2.7 and Fig. 2.8 show a time series of the same network metrics for the circ(1) trace with omnidirectional and directional antenna configurations, respectively. These tests have fewer dropouts than the circ(4) tests, due to being at a closer range. The SNR and throughput can clearly be seen as increasing as the slant range decreases, and decreasing as slant range increases.

Antenna pattern and aircraft symmetry: The Bat-4 is not symmetrical and parts of the aircraft can interfere with wireless communication. As previously mentioned, this is common in fixed wing aircraft and optimal antenna placement often depends on the intended environment and flight patterns for the aircraft. Additionally, a closer look at the antenna pattern specification sheet 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 UAS orientation and antenna pattern contribute to lower performance in the **coming towards** phase of the orbit, which is consistent with our data analysis. Next, we explore the impact of distance and orientation with regard to dropouts.

Dropout analysis: We define a **dropout period** as a period of at least one second in which the TCP throughput is zero. Dropout periods are typically caused by the SNR dropping below a certain threshold, which causes a temporary link failure in the radios. While dropouts are typical in wireless networks, especially at extended distances, we seek to further understand the dropouts based on distance and orientation.

Fig. 2.9 shows the percentage of time in dropouts for each orbit, along with the orientation phases of the orbits. We observe the **coming towards** phase of the orbit has higher percentage of dropout time than the **going away** phase for both antenna types, validating the qualitative observations from the previous subsection. The omnidirectional tests expe-

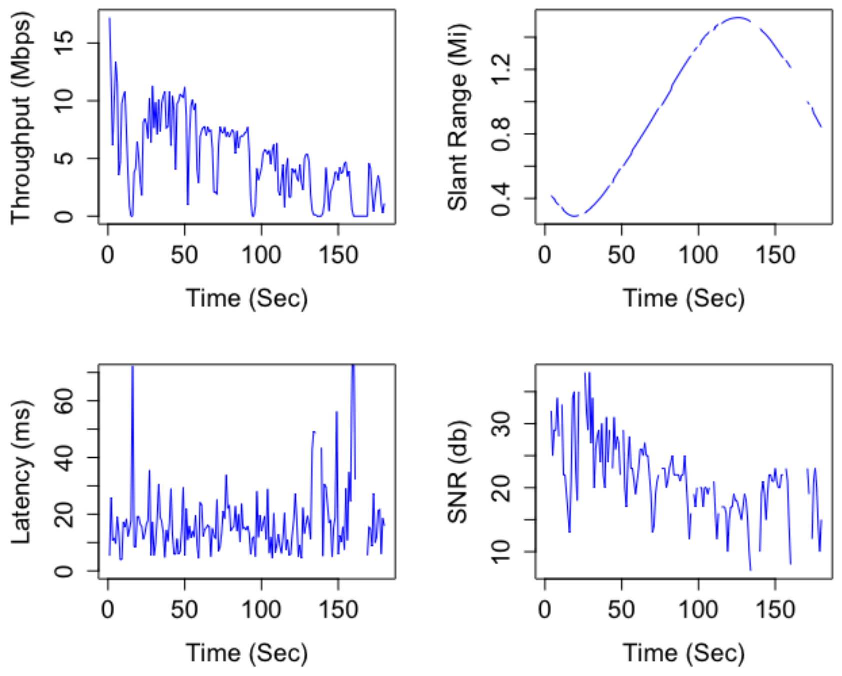


Figure 2.7. Omni circ(1) metrics

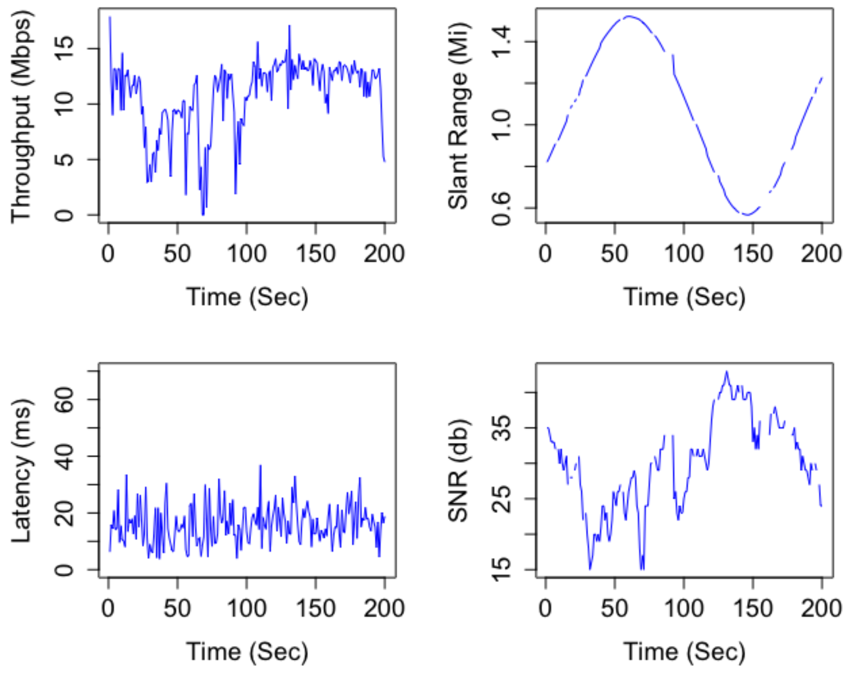


Figure 2.8. Directional circ(1) metrics

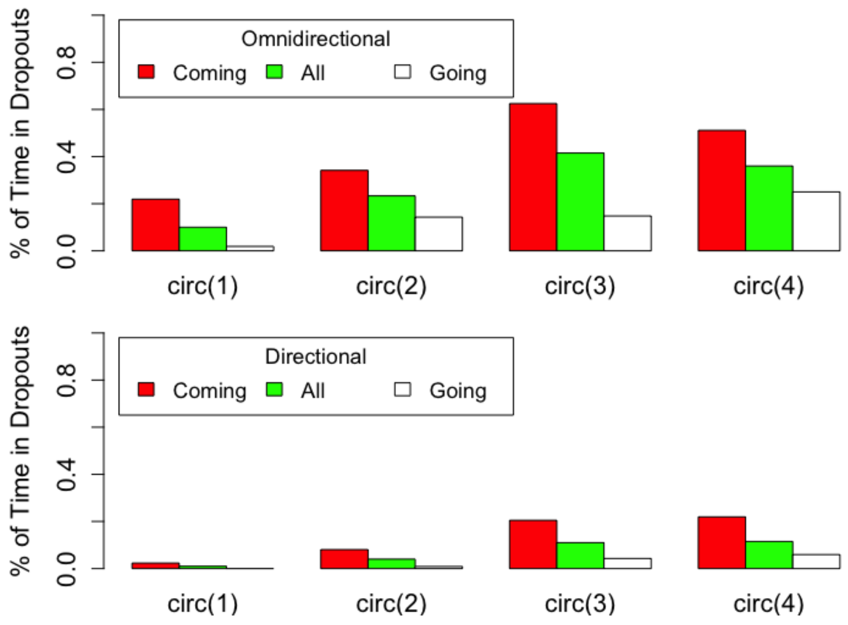


Figure 2.9. Percentage of time in dropouts

rience significantly higher dropouts than directional, as expected. The percentage of time in dropouts increases sharply from circ(1) to circ(2) and again from circ(2) to circ(3), as distance increases. More interestingly, the omnidirectional test exhibits a decrease in percentage of time in dropout when moving from the circ(3) to circ(4) flight pattern. Upon review, we determined that this can be attributed to the time spent in each orientation during these tests. In particular, the UAS is **going away** for only 44% of circ(3), compared to 58% of circ(4). Each of these tests completed a full circle orbit and small part of another orbit for each (to ensure completeness, given slight variance in UAS speed during orbit). Each test started in a different position in the circle, resulting in different times spent in each orientation due to the extra portion of an additional orbit flown. We also considered average dropout duration, and found it generally increases with distance, and is higher for the **coming towards** direction.

Throughput analysis: We show boxplots of the throughput for each distance and orientation in Fig. 2.10. While there is a large variation in throughput across the traces, there are some clear trends that are consistent with previous findings of performance differences, based on distance and orientation. We also analyzed SNR data, and it shows similar trends as throughput (SNR degrades with distance, and depends on UAS orientation).

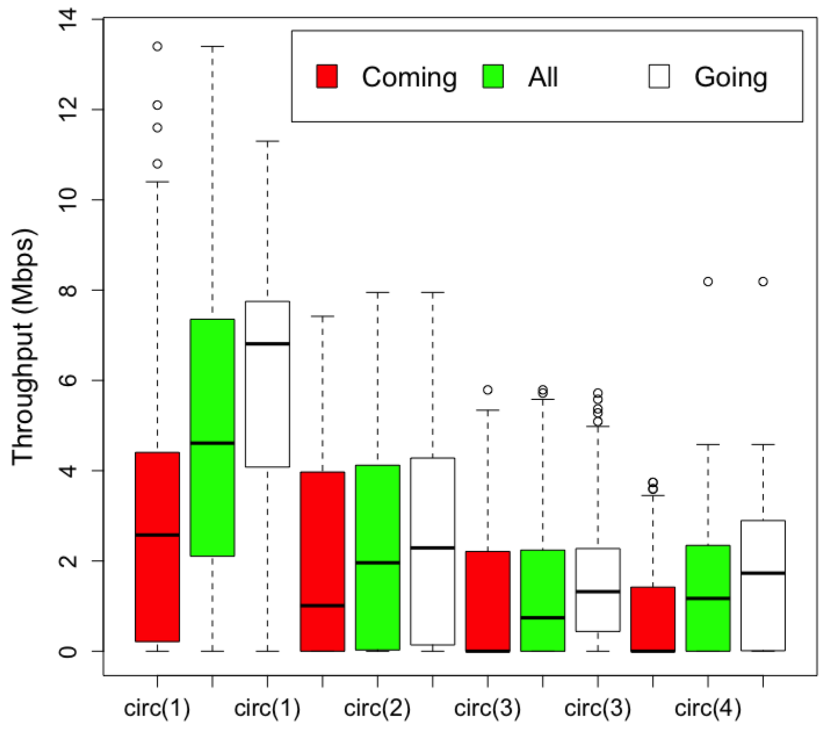


Figure 2.10. Omni throughput boxplots

Distance: We notice in Fig. 2.10 that throughput generally decreases as distance increases, as expected. 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 spent in the **coming towards** orientation, whereas circ(4) has additional time in the **going away** orientation, resulting in fewer dropouts.

Aircraft orientation: We also see in Fig. 2.10 that the third quartile in the **going away** phase is higher than in the **coming towards** phase across every trace. The first quartile is visibly higher for the **going away** phases in circ(1), circ(2), and circ(3), compared to the **coming towards** phases of those orbits. Due to the large number of dropouts, some of the quartile data is at or close to 0, making it difficult to directly compare.

Throughput time series analysis: We next investigate the correlation of throughput over time; this will indicate how much past bandwidth is a predictor of future bandwidth, with implications for different sensor data transmission algorithms. We use time series analysis testing to determine the correlation of the throughput data over time for each circle trace, with circ(1) and circ(4) omnidirectional and directional results shown in Fig. 2.11. The dashed lines represent the 95% confidence interval for an uncorrelated process; in other words, if the samples were uncorrelated over time, we would expect the sample autocorrelation at each lag to be inside the indicated bands with 95% confidence.

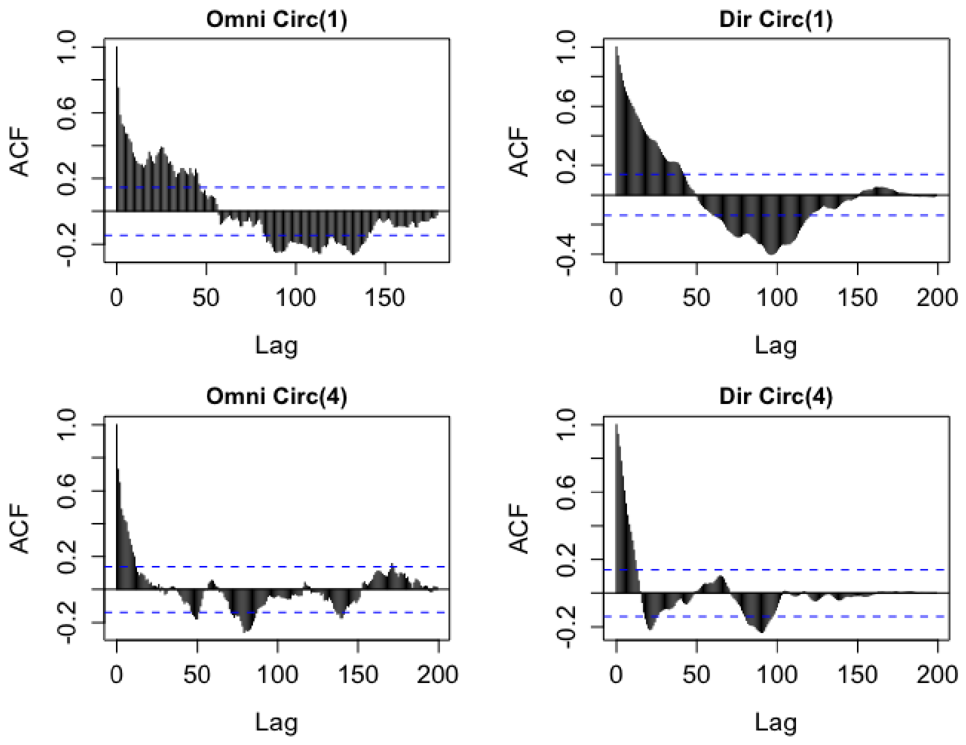
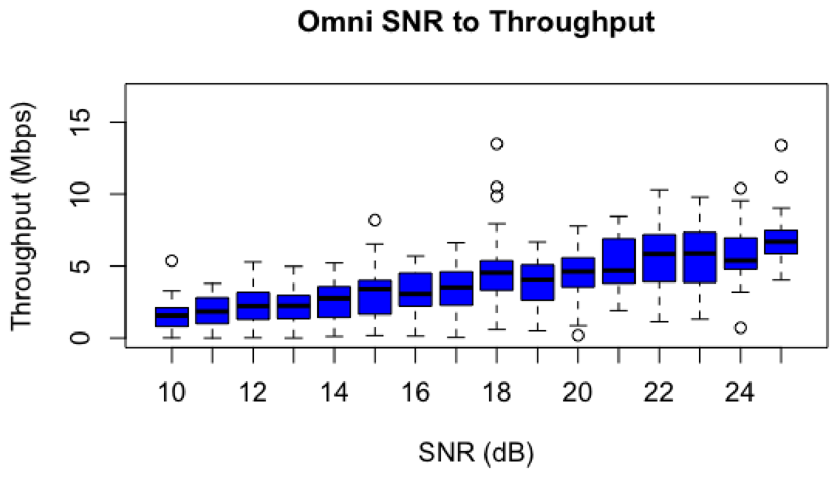


Figure 2.11. Autocorrelation of throughput

These tests show the throughput values are correlated over time. This is because the UAS moves in a seasonal pattern through the circle over time, resulting in performance differences based on the orientation and position of the aircraft. Interestingly, the autocorrelation plots also reveal the differences in the throughput due to the orientation of the UAS. In particular, since the circular orbits take roughly 160-180 seconds to complete, we notice the anticorrelation of data occurring about halfway through, as a trend for all graphs. This represents the time-lag between the **coming towards** and **going away** orientations, agreeing with the results presented earlier. Additionally, the throughput is more closely correlated for longer lags at shorter distances and less correlated as the aircraft moves further away, due to increased throughput variability at further distances. Both trends agree with our earlier findings.



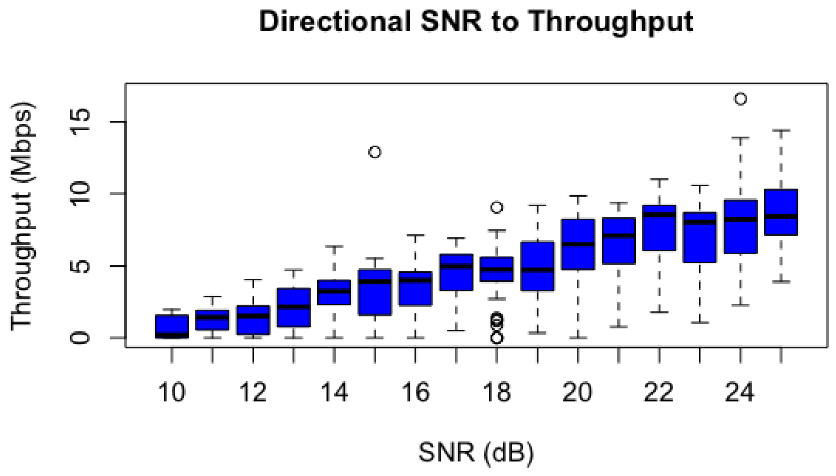


Figure 2.12. Omni and directional SNR to throughput

SNR throughput analysis: Fig. 2.12 shows a comparison of the throughput in relation to SNR for both antenna configurations. The results show a general increase in throughput as SNR increases, as expected. The throughput ranges for each SNR value are relatively similar for each antenna configuration, as expected. The overall throughput for directional is higher than omnidirectional because the flight is in a higher SNR range at the same distance, due to increased signal strength with a higher gain antenna.

2.4.2 CA dataset analysis

Our CA data collection culminated in 12 TCP traces and 35 UDP traces across 3 days of flight testing, for a total of roughly 5 hours of data collection time. We collected more UDP traces because more data is required to calculate the available throughput of the communication channel (as described below). TCP transmission automatically adjusts to the expected available throughput by using the network statistics (throughput, loss, delay, etc) and increasing or throttling the sender rate. UDP transmission rate, on the other hand, is determined by the sender. If the UDP sender rate is too high and exceeds the available throughput, then data can be lost or delayed. However, if the available throughput exceeds the UDP sender rate, then there is an opportunity cost where more data could have been sent. Generally, to find the available throughput, multiple UDP transmission rates should be tested. The available throughput is: $Throughput = transmissionRate \times (1 - lossRate)$. However, if too much data is sent, then the congestion can cause this calculation to not be exact, which is why we include multiple UDP traces. Our process for calculating available throughput includes UDP measurements conducted at multiple different transmission rates (several loops were flown in each orbit, and a different transmission rate used in each loop), and we consider the effective throughput seen by the receiver when the sender transmitted at a rate that saturated the link.

UDP test datasets: We focus our analysis on the UDP tests, since we have previously described the TCP tests from our FL datasets, and the TCP data from the CA testing was qualitatively similar.

We first explore **UDP loss rates**, based on distance. Fig. 2.13 (left) shows loss rates for the CA data grouped by distance ranges with a UDP transmission rate of 5.0 Mbps. We see the loss rate increases with distance, as expected. More interestingly, the right figure shows the UAS orientation affects network performance, revealing higher UDP loss rates in the **coming towards** orientation. This is similar behavior to the FL dataset, with fixed wing UAS, but surprising that the multirotor UAS exhibits the same performance, since it is symmetrical. Upon further inspection, the multirotor UAS tilts slightly in the direction it is facing and thus has some obstruction while **coming towards** (shown in Fig 2.14).

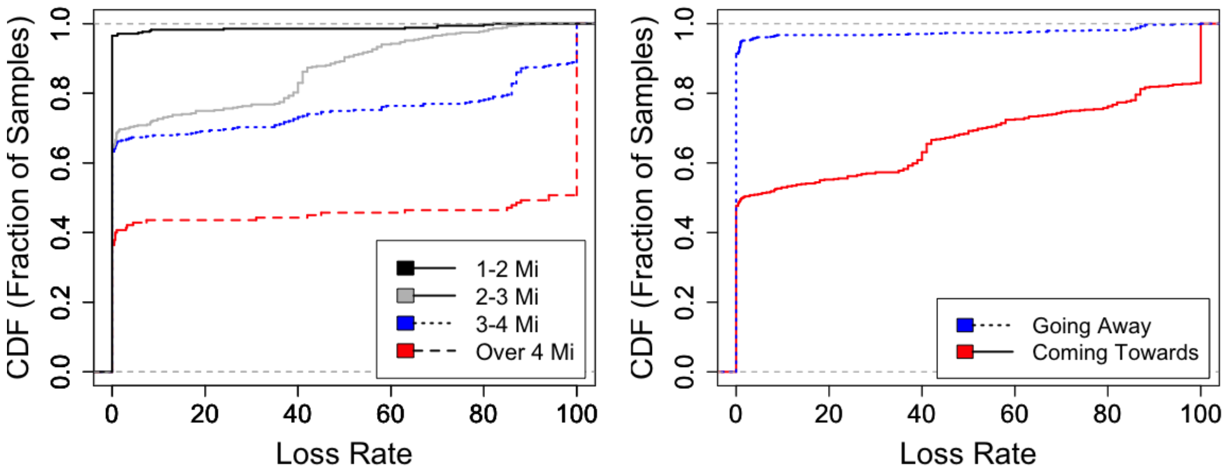


Figure 2.13. UDP loss with distance (left), and orientation (right), **CA**

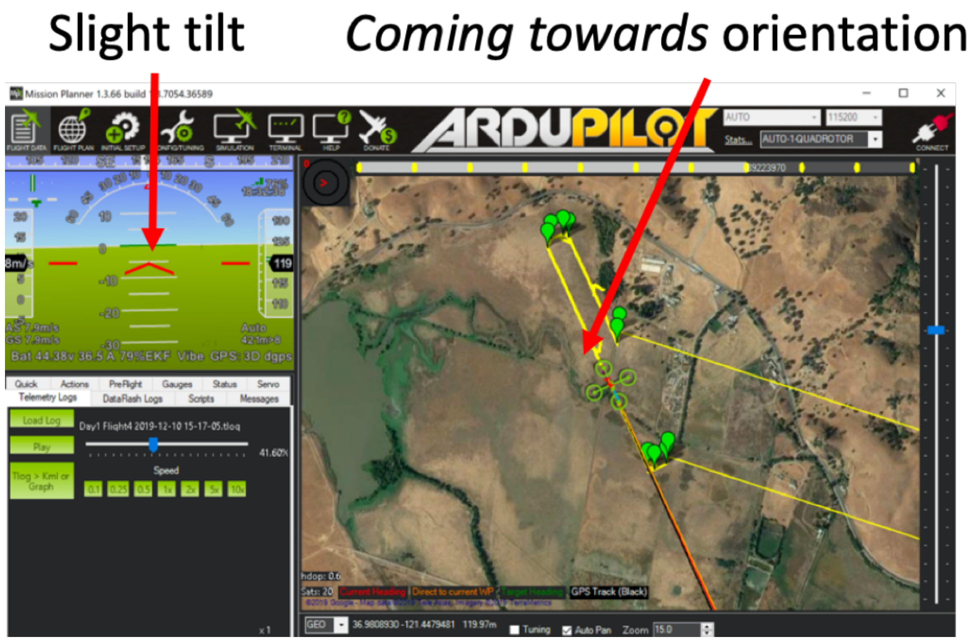


Figure 2.14. UAS controlled by Mission Planner software, showing UAS attitude is slightly titlted, leading to performance decrease while the UAS is *coming towards* the GCS.

In Fig. 2.13, notice that loss rates in the range of over 4 miles are too high to support most usable sensor data applications, while performance at a range of under a mile was typically abundantly good, such that simple transmission algorithms in this range are sufficient for data transmission. Hence, we focus our analysis on data collected in **CA1-2** (see Fig. 2.4).

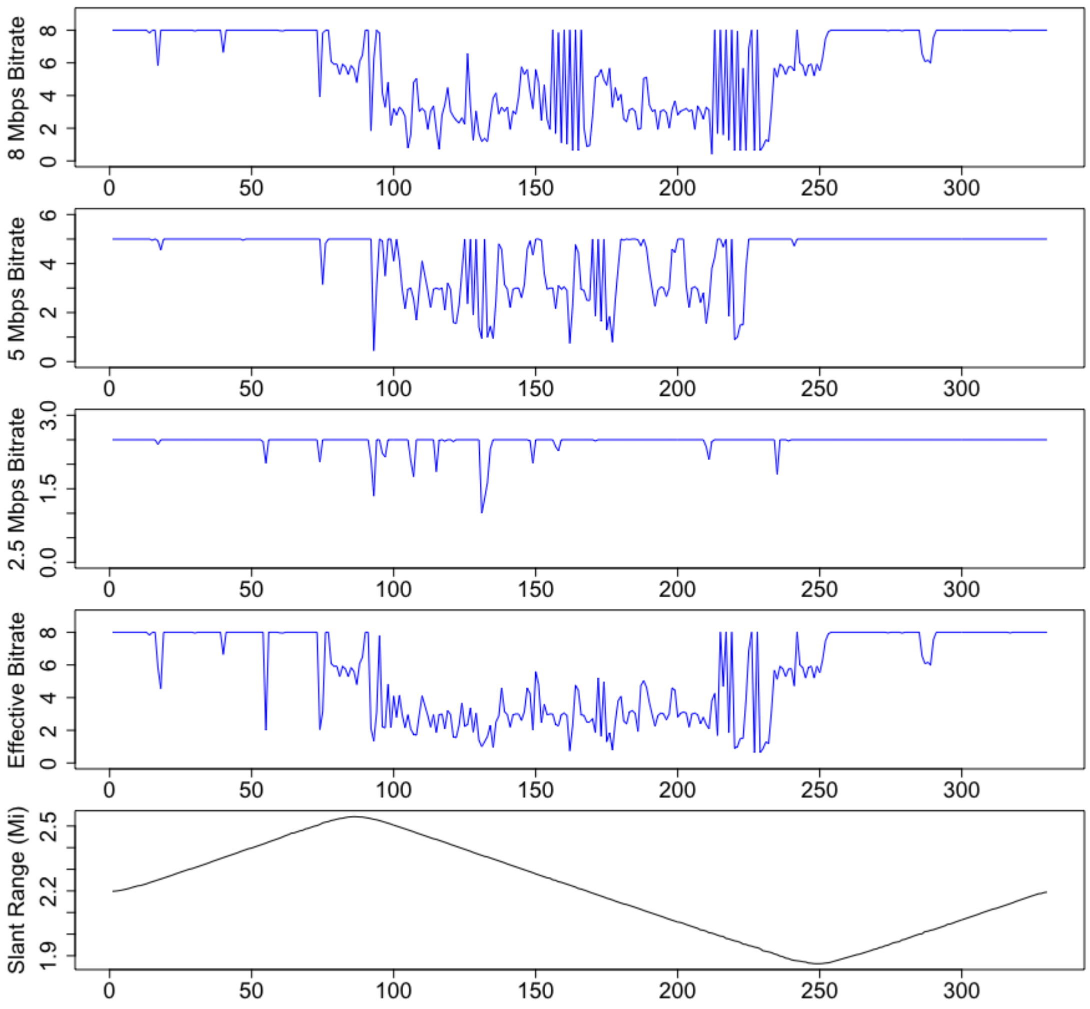


Figure 2.15. UDP throughput for **CA1**

Deeper analysis of UDP throughput: We test CA1 and CA2 with several UDP sending rates, and combine three tests with transmission rates of 2.5, 5.0, and 8.0 Mbps, respectively. Fig. 2.15 and Fig. 2.16 show the effective throughput for sending at each of these rates for CA1 and CA2, respectively. The top plot shows sending with the highest transmission rate (8.0 Mbps) and the middle plot shows sending with the lowest transmission rate (2.5 Mbps), with a transmission rate of 5.0 Mbps in between those two plots. We see more variability from UDP losses as the transmission rate is higher, as expected. Further, we combine the three traces by taking the most link-saturated rate (labeled **Effective Bitrate**

and shown as the second plot from the bottom). Finally, the slant range of the UAS from the GCS is shown on the bottom plot. We notice the data for the CA2 location, which is further distances than CA1, is slightly more variable (especially in the 8 Mbps UDP transmission rate). This is expected, as the distance increase causes a decrease in wireless performance. The combination of the UDP send rates into the effective UDP throughput is what we use for our UDP test traces with Chimera in §4.

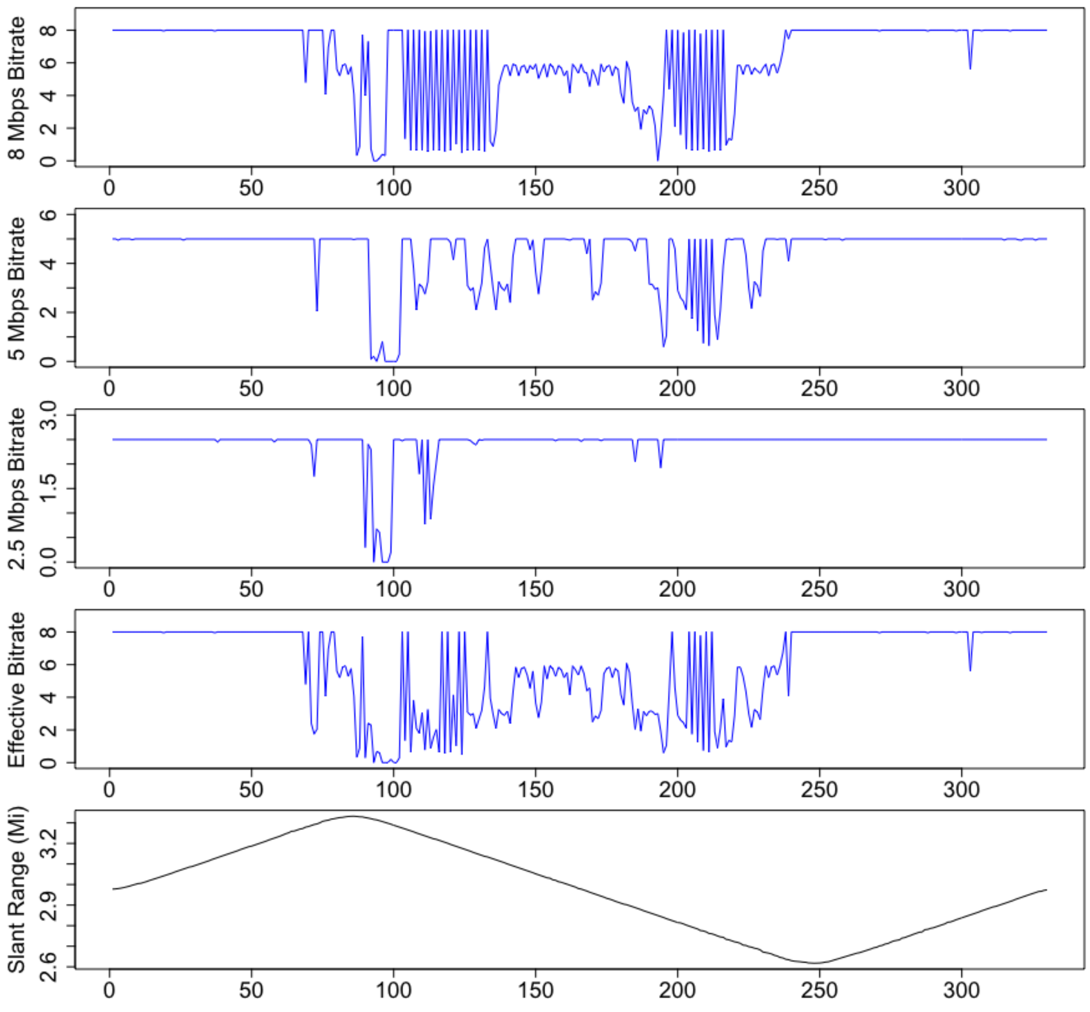


Figure 2.16. UDP throughput for **CA2**

2.4.3 Flight dataset takeaways

Our flight tests were executed using both multirotor and fixed wing UAS at two locations across the US. These tests were unique and present relatively unexplored aspects of UAS flight beyond VLOS, due to the challenges in collecting such data, but representing a growing area of interest, as discussed in §2.2. The datasets show the dependence of UAS throughput on the UAS flight plan, for both multirotor and fixed wing UAS types. The throughput degrades with distance, as expected. More interestingly, the UAS throughput depends on the UAS flight orientation. This is especially interesting that it affects both UAS types, albeit for different reasons, as we discussed. The observation that the UAS flight path significantly impacts throughput motivates the rest of our research as we seek to exploit this information to design new UAS sensor data transmission algorithms that are optimized for UAS flight. Additionally, the flight test datasets provide traces from which we can further analyze and design throughput prediction models with (§4), and use for simulation and emulation testing (in both §3 and §4).

2.5 Related work

There has been much recent interest in using UAS to extend Internet connectivity to remote locations [34], [44], [45]. The primary focus of these works is optimal positioning of the UAS to best serve the area [44], and how to best support LTE base station functionality on UAS [34]. There has been work with a focus on 802.11 fixed wing [30], [43] and multirotor [29], [31], [33], [35] UAS networking, but at limited distances (less than 0.25 miles). Additionally, commercial In-Flight Communication (IFC) is characterized in [61], showing there is significant packet loss and throughput variation in such settings. A recent workshop paper [41] conducted a preliminary investigation of UAS for video streaming. Limited experiments with a multirotor and WiFi over short distances were provided. Finally, different types of Tactical Radios are tested with different transport protocols on the ground in [47]. In contrast to these works, our focus is on UAS networking for applications that require aerial surveillance coverage, often where it is not advisable to step foot close to the area being inspected. For these scenarios, it is important to enable acceptable performance over extended distances, and when the location cannot be optimized for connectivity. We focus on Tactical Radios rather than LTE settings, providing extended reach and no reliance on pre-existing infrastructure.

2.6 Conclusion

Our experiments explored long range measurements with both fixed wing and multirotor UAS using Tactical Radios. Our data is collected at multiple locations across the US, and presents both TCP and UDP transport data. Further, we have analyzed the flight test data with traces at different distance ranges and with different corresponding antenna configurations. In our analysis, we found that the network performance varies over the duration of the flight with both **distance** and **plane orientation**, making networking with dynamic UAS flight challenging, but also providing opportunity to potentially leverage the flight path in UAS networking applications. Our results can be used to better tailor algorithms for delivery of critical aerial surveillance information, and improve the overall experience for end users (as shown in our work presented in the following chapters). One implication is that we need to not only (i) consider the flight's distance, but also relative orientation; and (ii) we need to explicitly consider dropouts in the application design. Our tests explore performance all the way to the edge of connectivity. Even in this worst case at the edge of connectivity, sensor data (e.g. video) delivery at certain resolutions may still be possible, but dropouts must be considered.

3. OPTIMIZING QUALITY OF EXPERIENCE FOR LONG-RANGE UAS VIDEO STREAMING

3.1 Introduction

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). An example of this type of scenario is shown in 3.1. The figure represents a military ground operator communicating with a UAS in order to survey and gather insight about an area with a video sensor. This type of scenario is also quite relevant to disaster response [2], [8], [13], search and rescue [14], law enforcement[2], agriculture [2], railroad and pipeline inspection [2], [10], and more [2], [11], [15].

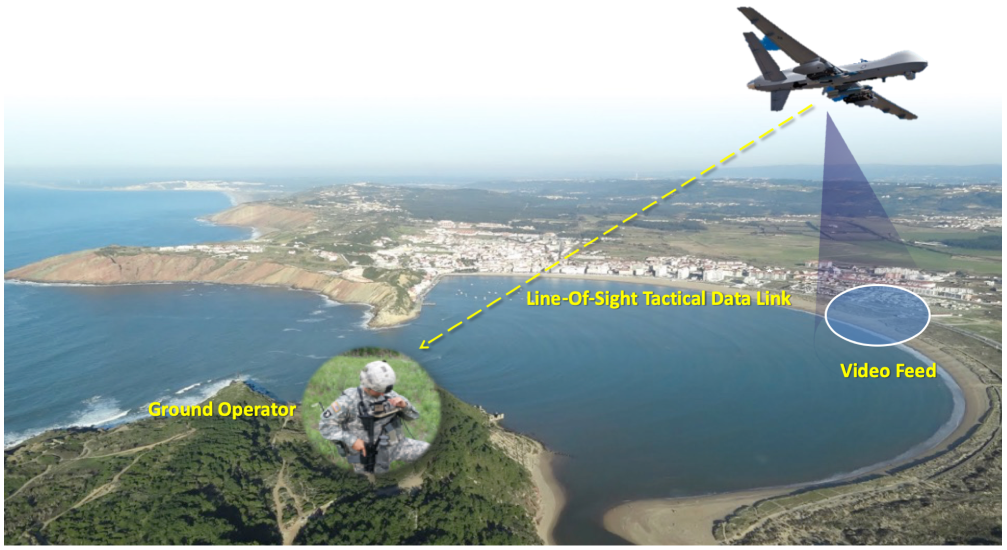


Figure 3.1. UAS video transmission

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 [62], [63], where it is feasible to optimally position UAS to ensure the best connectivity. As we discussed in §1, today's civilian use of UAS in the United States

(US) is typically limited to distances that require visual line of sight [20] (typically, 1 Km). However, we also noted that there is interest in extending the range of UAS networking, and regulations are being put into place to support such initiatives [1], [10], [11].

In this chapter, 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 chapter, we answer the above questions and make the following contributions [48].

First, we utilize the data analysis of UAS networks based on real-world data from UAS flights at long-range distances (beyond VLOS) presented in §2. We use the observations gleaned in the previous chapter but also take a deeper look at the specific measurement results pertaining to video streaming. We do so in order to better understand the applicability for such applications in dynamic long-range UAS environments. 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.

Second, motivated by the observations above and from §2, we present Proteus, a system for video streaming in UAS settings. Proteus leverages Adaptive Bit Rate (ABR) algorithms [36], [37], [40], [49]–[51], 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 [15], [41]. 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 streaming [39], [50], [51]. Unfortunately, we illustrate that a direct application of a representative and widely studied ABR algorithm based on a control theoretic approach [50], 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 the introduction of 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 [52]; 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 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) [36], [37], [50], 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.

[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 [50], following convention in the video streaming community

3.2 Motivating measurements of UAS networks

In this section, we present measurements from real-world UAS flights, building upon our measurements in §2, with a focus on how the data will function for video streaming applications. 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.

3.2.1 UAS flight test setup

We used the FL test data and radios as we described in §2: flying a Bat-4 [64], a representative fixed wing UAS in terms of size, weight, and speed, and appropriate for the activities described in §3.1. Fixed wing UAS can benefit video streaming application, since the video streaming applications we focus on benefit from speed and longer endurance.

Recall the flight common flight terms **Distance, Slant Range, and Altitude**, relative to the Ground Control Station (GCS), shown in Fig. 2.2 from §2, and the different UAS orientation terms of **coming towards** (to the GCS) and **going away** (from the GCS).

For UAS wireless connectivity, recall that our FL testing used two Persistent Systems MPU4 Tactical Radios [47], [58] 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. For video streaming at long-range distances, we focus on the omnidirectional ground antenna test configurations since they are 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 and we also provide test results for directional antennas in §3.4 in order to show the flexibility and broad applicability of our work. As previously discussed (§2), we collected TCP throughput using iPerf [60] and used pings to record latency and loss rate.

As discussed in §2, circular orbits around a point are useful for recurring video coverage of an area. Recall that we refer to the data collected from circle orbits as **circ(k)**, with

$k \in \{1, 2, \ldots, 7\}$, depending on the distance from the GCS to the center of the circle. Our work in this chapter focuses on omnidirectional configurations, for optimal flexibility, but we also consider comparisons directional antenna configurations, as previously described. Thus, we utilize the distances up to circ(4). 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).

3.2.2 Findings and implications for UAS video streaming

Recall Fig. 2.5 shows time series plots of the UAS slant range and network metrics (throughput, latency, and SNR) for the circ(4) trace. This figure shows 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). Recall that 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 on network performance. We saw in §2 that this impact is indeed present, and will further quantify the differences in network aspects related to video streaming (especially dropouts) 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 also explore additional measurements (relative to §2), in order to provide a better understanding of the flight test data, and how it relates to video streaming. To accomplish this, we next measure consecutive periods of loss, which is an indication of no data going through for a certain period. We also measure median and maximum loss duration, as well as average dropout duration, as each of these measurements will have an impact on our video streaming performance (with specific impacts on the buffer, since the video buffer needs to be large enough to sustain a prolonged dropout period). Fig. 3.2 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).

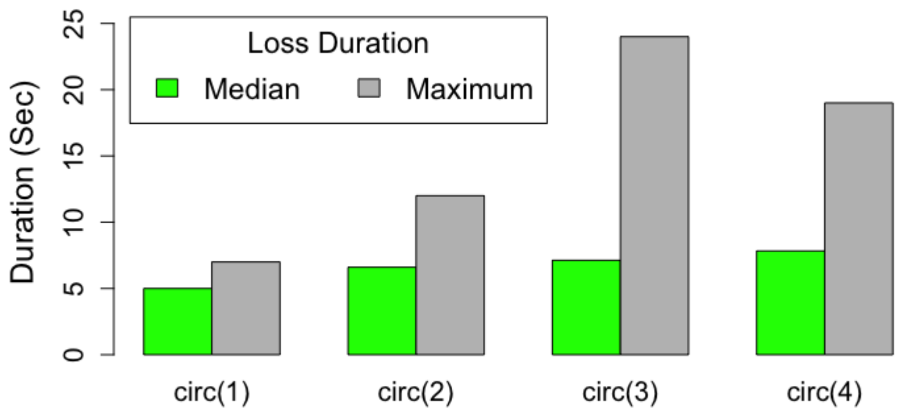


Figure 3.2. Median and maximum loss duration with pings

We also measured ping loss rates to compare to the TCP dropout rate that we previously measured in §2. The results show the clear trend of going away exhibiting a lower loss rate than coming towards, agreeing with our previous assessments.

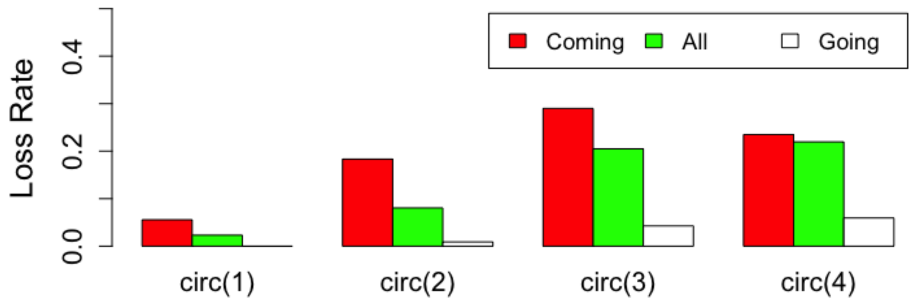


Figure 3.3. Loss rate with pings

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.

Fig. 3.4 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. 3.4 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 §3.3.

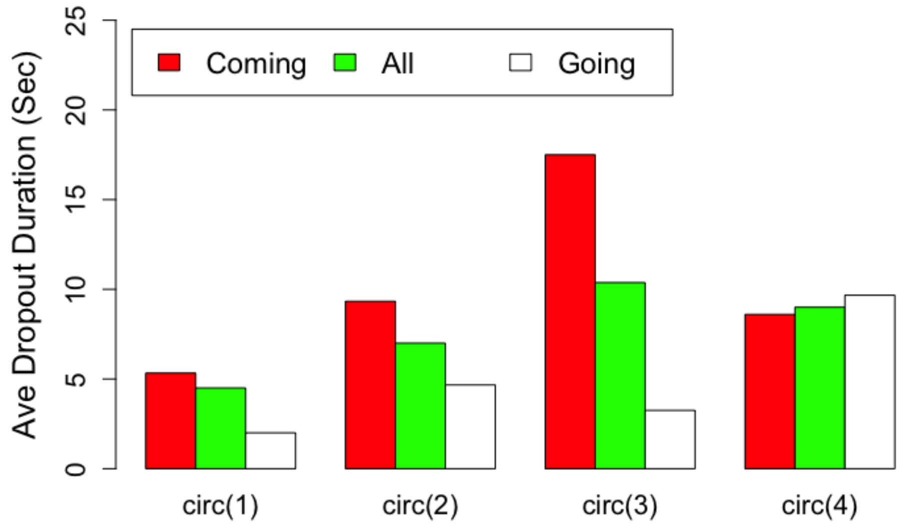


Figure 3.4. Average dropout duration

Correlation of throughput with flight path presents opportunities to improve video streaming. Recall Fig. 2.10 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.

Fig. 2.10 showed that throughput is generally higher for the *going away* orientation, compared to *coming towards*, consistent with earlier measurements. As we discussed in §2, we believe this is because fixed wing UAS are not symmetrical and parts of the aircraft can 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, recall that a closer look at the representative radiation patterns on the specification sheet [65] 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 UAS orientation and antenna pattern contribute to lower 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 §3.3, we will explore how Proteus leverages these correlations in its design.

3.3 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 §2.3.

3.3.1 Proteus design rationale

Solutions for video streaming in UAS settings have multiple design points. At one end of the spectrum, one could target ultra-low sub-second latencies. However, our measurements in §2 indicate that this design point 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. 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 are acceptable in the vast majority of military and civilian applications [7], [66]–[68], especially when the delay extends the mission flight range. For instance, in disaster response settings [2], [13], 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 scenarios 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 [69], which allows use of rate adaptation mechanisms on top of UDP. We next discuss why existing ABR algorithms are inadequate, and present Proteus' approach.

3.3.2 Need for new ABR approaches for UAS settings

While many ABR algorithms have been developed in recent years [36], [37], [40], [49]–[51], [70], 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 [50], a widely used and representative ABR algorithm. Our insights can benefit other ABR algorithms as well, as we discuss in §3.5. MPC uses a combination of future throughput prediction and buffer occupancy to select chunk bitrates [50].[2] Prior to downloading each chunk, the MPC algorithm selects the next bitrate by optimizing a composite metric (more formally defined in §3.3.3) 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 [52].

We tested MPC via an emulated test-bed (see §3.4 for experimental setup details) with throughput traces gathered from our UAS flights. Fig. 3.5 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 §3.4.

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

[2]↑We use the term MPC to refer to the conservative version of the algorithm (referred to as *RobustMPC* in [50]) which reduces predicted throughput by using a discount factor based on the maximum error in throughput predictions experienced over the last few chunks.

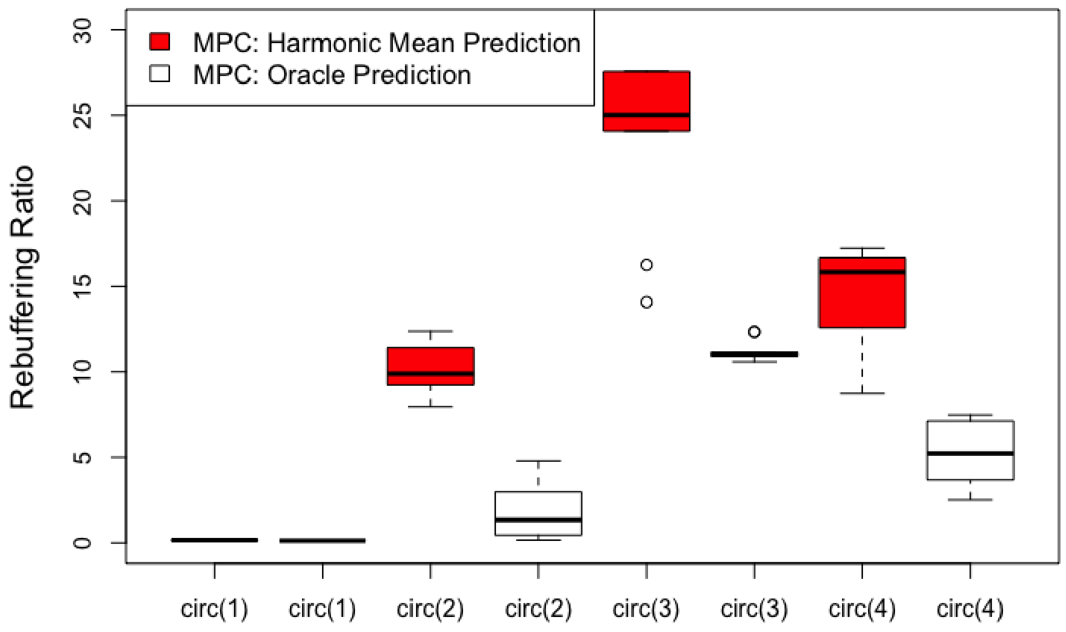


Figure 3.5. MPC rebuffering ratio for circ(4) with a harmonic mean predictor and an Oracle predictor

3.3.3 Proteus design details

In this section, we discuss our new algorithm design, Proteus, which overcomes previously discussed challenges by (i) 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 [52], 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[36], [37], [50], defined as:

$$QoE(\mathrm{i},\mathrm{j}) = \sum_{n=\mathrm{i}}^{\mathrm{j}} R_n - \mu \sum_{n=\mathrm{i}}^{\mathrm{j}} T_n - \lambda \sum_{n=\mathrm{i}}^{\mathrm{j}-1} |(R_{n+1}) - (R_n)|. \tag{3.1}$$

Here, $QoE(\mathrm{i},\mathrm{j})$ 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:

$$QoE_b = QoE(\mathrm{i}, \mathrm{i} + W - 1) + \gamma\epsilon(b). \tag{3.2}$$

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 a factor that 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. (3.1). 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:

$$\epsilon(b) = \frac{\bar{b}^2 - (\min(b, 2\bar{b}) - \bar{b})^2}{\bar{b}^2}, \tag{3.3}$$

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. 3.6 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.

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 bit rate 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.

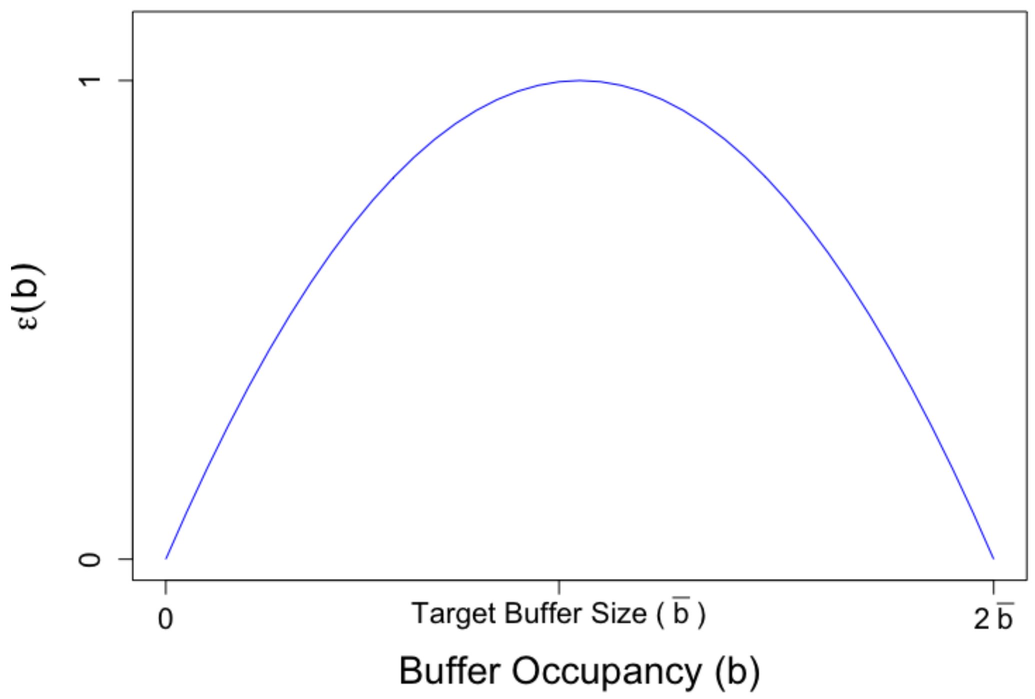


Figure 3.6. An illustration of the terminal cost $\epsilon(b)$ from Eq. 3.3, representing the reward obtained by having a buffer occupancy b.

We next discuss how the $\bar{b}$ 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 $\bar{b}$ 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 $\bar{b}$ and α for each orientation (*coming towards* and *going away*).

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 [36], [37], [50]). To illustrate our simulation, Fig.

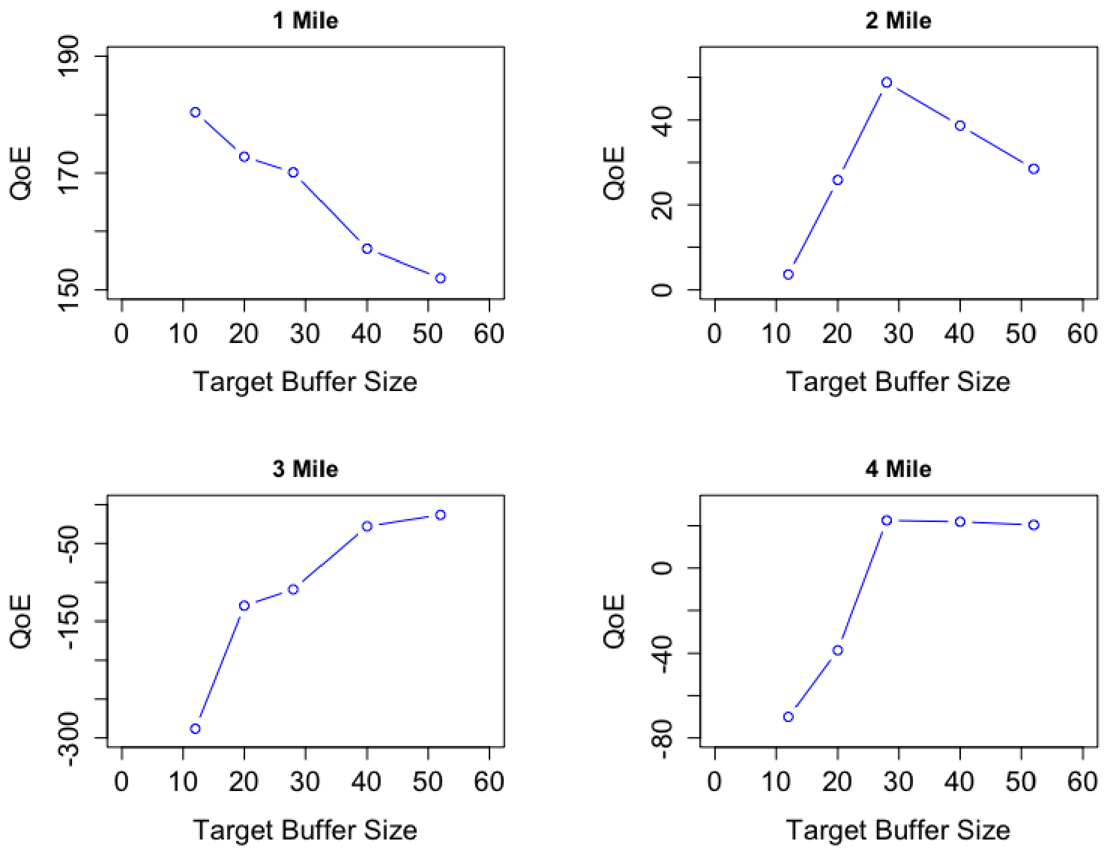


Figure 3.7. Target buffer size tuning for the circular orbits

3.7 shows how the achieved QoE varies based on the $\bar{b}$ parameter, while keeping α fixed at 3. The figure shows that in general larger distances require a larger target buffer size $\bar{b}$. This is due to higher dropouts and lower throughput as distance increases. We can see that low values of $\bar{b}$ 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 §3.4. As an example, one can see the optimal selection for Proteus for circ(3) is $\bar{b} = 52$ (with $\alpha = 3$), although with Proteus-Orient, the optimal configuration becomes $\alpha = 3$ and $\bar{b} = 28$ for *going away*, and $\alpha = 5$ and $\bar{b} = 52$ for *coming towards*. This aligns with our flight measurements, 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 the 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 §3.2, and set its parameters based on those learnt from the *Circle* trace; we will present those results in the next section.

3.4 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.

3.4.1 Emulation methodology

We use Mahimahi [71] 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 [72]. We chose this video because it has been extensively used in prior work [36], [37], [40], 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 [50] presented earlier (Eq. (3.1), with i=1, and j = N indicating that performance is measured over all video chunks). Note that although Proteus optimizes the modified metric in Eq. (3.2) 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 the video performance.

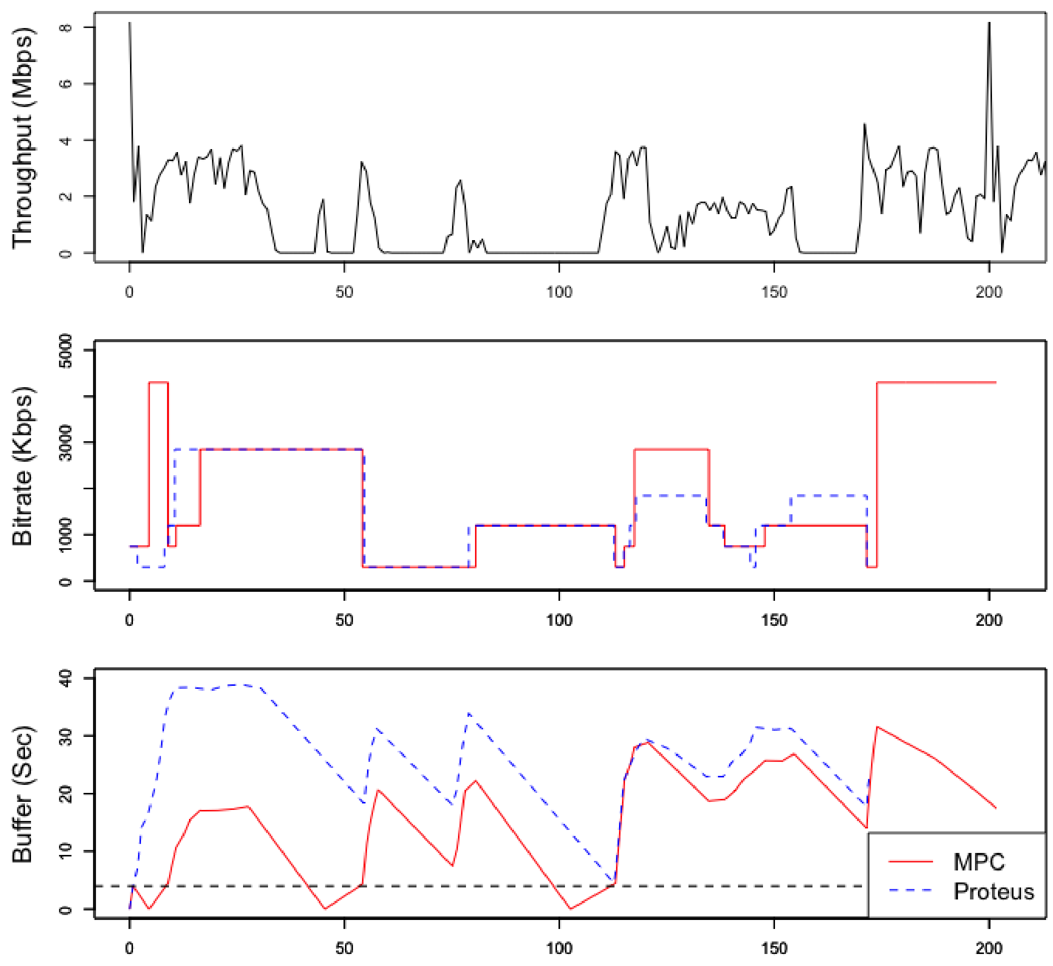


Figure 3.8. Proteus illustration for circ(4)

3.4.2 Benefits of Proteus

Fig. 3.8 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 §3.3), 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.

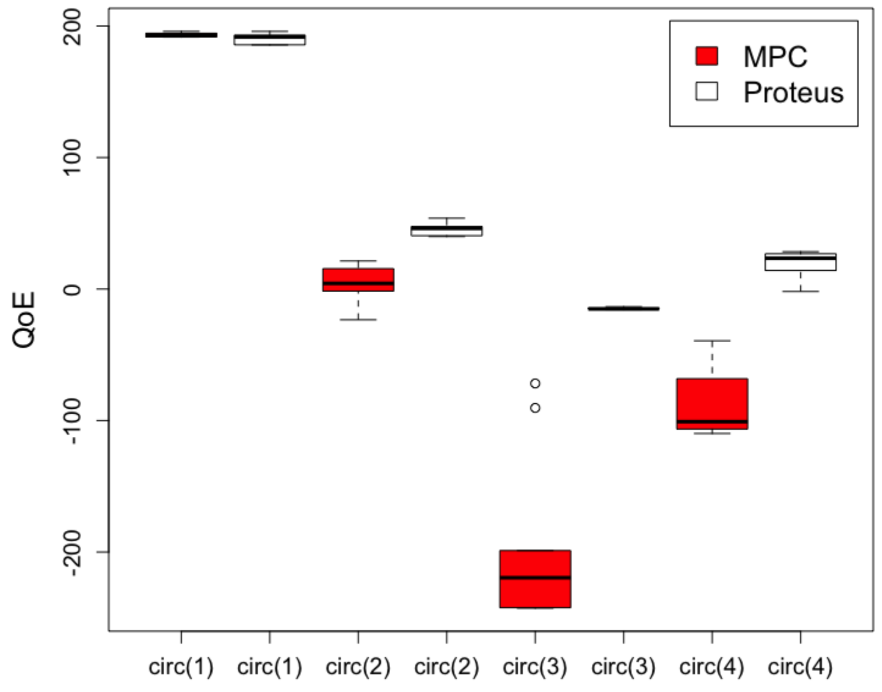


Figure 3.9. QoE benefits with Proteus

Fig. 3.9 compares MPC with Proteus for the different circle traces. Each circle trace uses different insurance parameters, as described in §3.3. 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.

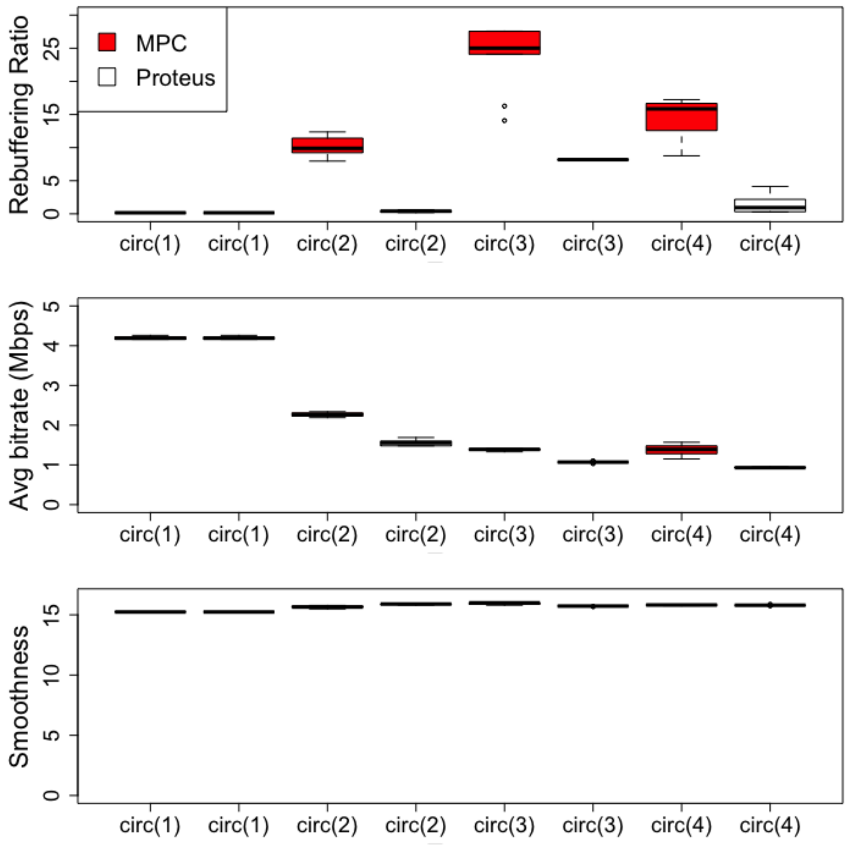


Figure 3.10. Breakdown of individual video delivery metrics

Fig. 3.10 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.

3.4.3 Benefits of Proteus-Orient

Fig. 3.11 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 duration. Finally, we omit circ(1) because Proteus did not improve performance, due to less dropouts at this distance.

3.4.4 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. The *Line* trace is a straight line from a distance of 2.75 miles from the GCS to 0.75 miles. We use $b = \bar{b}$ 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*

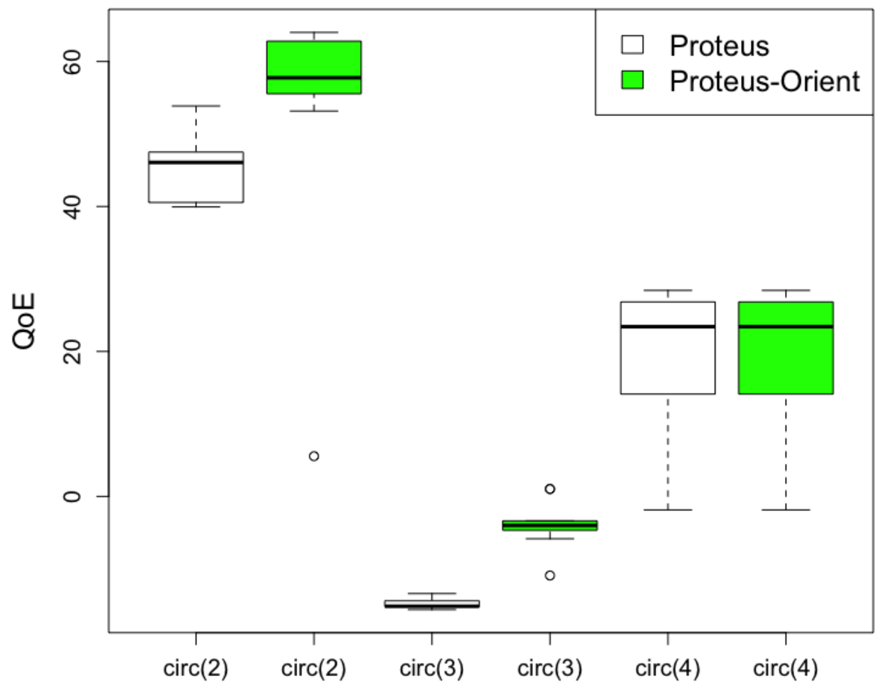


Figure 3.11. Benefits of considering orientation

scheme, which determines the best static $b = \bar{b}$ and α parameters for the *Line* trace. Fig. 3.12 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*.

3.4.5 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 next evaluate whether Proteus benefits persist if better throughput prediction methods are used. We consider two such predictors: Hidden Markov Model (HMM) and an Oracle.

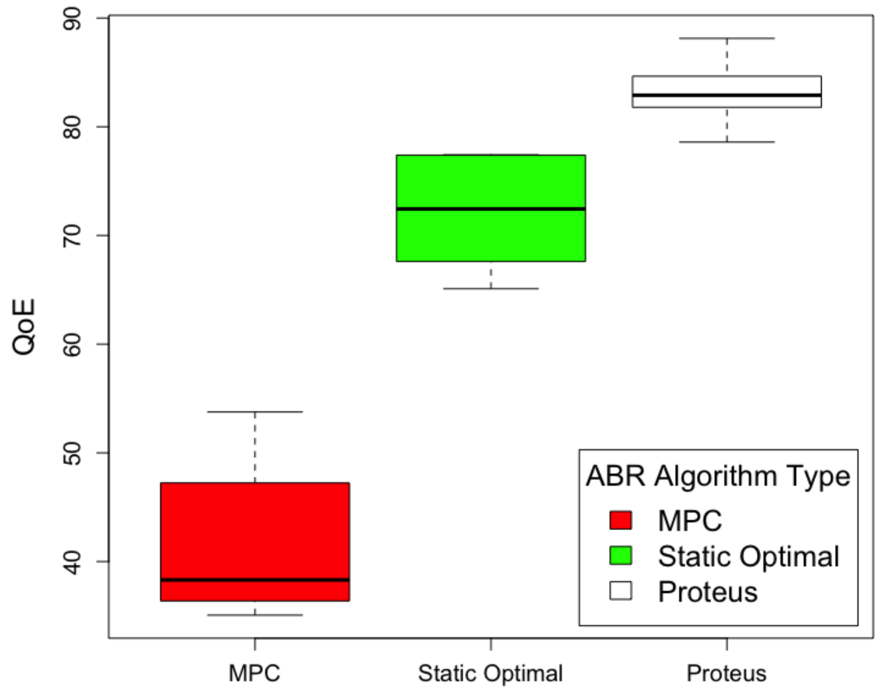


Figure 3.12. Proteus performance when parameters learned from one trace are applied in another trace and dynamically vary with distance

HMM: Recent work [38] 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, and -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 §3.3. We next evaluate both MPC and Proteus with an Oracle that knows the exact throughput information for the duration of the look-ahead window.

Fig. 3.13 shows that Proteus still outperforms MPC in the QoE metric, even with perfect Oracle throughput prediction. Fig. 3.14 shows that Proteus significantly reduces the rebuffering ratio while only slightly reducing the average bitrate.

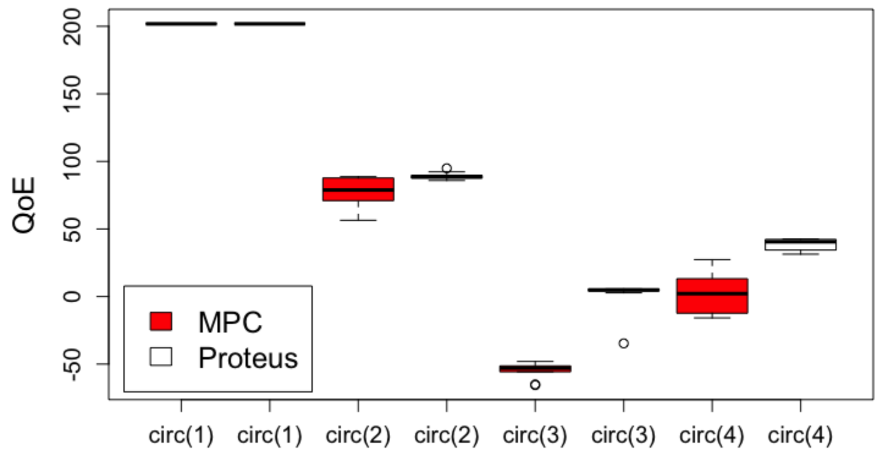


Figure 3.13. QoE benefits with Proteus (Oracle predictor)

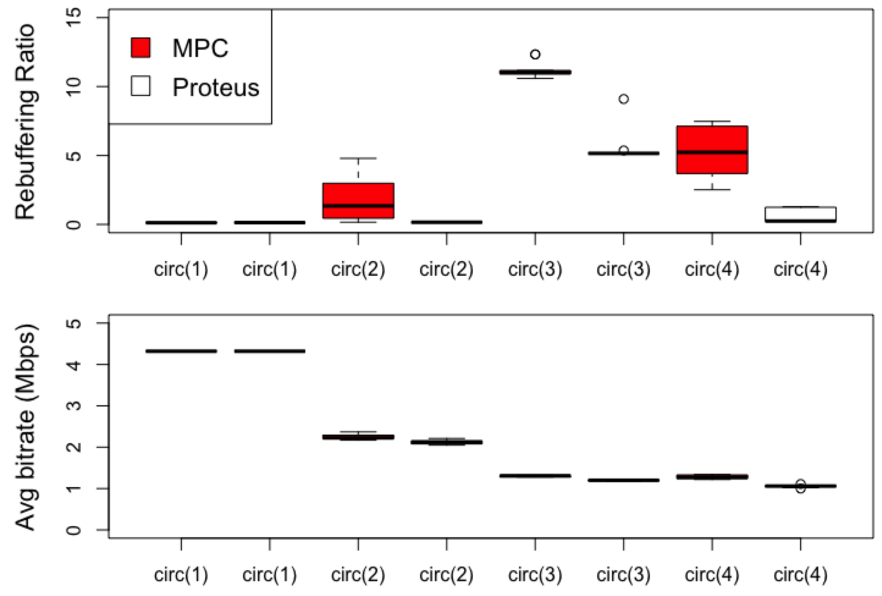


Figure 3.14. Oracle predictor performance measurements

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, as shown in Table 3.1. These results show there is still significant improvement over MPC by using Proteus, even with smaller buffer sizes.

Table 3.1. Benefits of Proteus with the client buffer size lowered to 30 seconds

Trace	**MPC**	**Proteus**
circ(2)	3.06	15.28
circ(3)	-194.45	-92.61
circ(4)	-59.27	-41.22

3.5 Related work

UAS networking: A recent workshop paper [41] conducted a preliminary investigation of UAS for video transmission using ABR with content-based compression. Another paper [15] details optimal algorithms to improve sports surveillance with UAS. These works present limited experiments with multirotor UAS and WiFi over short distances, and do not consider dropouts, which is an important challenge in our work as we explore ABR using flight test data from long-range distances.

Several efforts are underway to use UAS to augment Internet connectivity in remote areas [34], [44], [45], [62], [63]. In such settings, there is freedom to position UAS to best serve the area [44]. There has been work with a focus on 802.11 fixed wing [30], [43] and multirotor [31], [33], [35] UAS networking, but at limited distances. Tactical Radios are tested with different transport protocols on the ground in [47], but not in the air and 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 [73] uses measurement-based methods to adjust the video bitrate for video streaming. This work and the ABR algorithms [36]–[40] have been designed for traditional Internet environments and do not account for the challenges of dynamic long-range UAS flights. Recently, Elephanta [40] 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 [36] 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 the 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 [37] uses reinforcement learning to select bitrates. We are unable to evaluate Pensieve [37] 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 [37], and buffer-based approaches [39], [49] (e.g., the minimum and target buffer thresholds in [39] can be configured in a manner informed by the UAS flight path).

3.6 Conclusion

In this chapter, we have taken a key step towards enabling UAS video streaming at long-range distances [48], with three contributions.

First, we built upon the analysis of data collected from real-world UAS flight tests in §2 by considering additional analysis as it relates to video streaming. We showed 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 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.

4. CHIMERA: EXPLOITING UAS FLIGHT PATH INFORMATION TO OPTIMIZE HETEROGENEOUS DATA TRANSMISSION

4.1 Introduction

UAS are increasingly used to perform sensing and data-gathering in a variety of scenarios, due to their ability to go to areas where humans cannot, and gain vantage points and coverage with sensors that are not possible from the ground [1]–[6]. The types and capabilities of sensors, and options for mounting on UAS, have increased to provide many options to gain insight from an aerial viewpoint [4], [7]–[9]. Furthermore, the data transmitted from these sensors has diverse network requirements; for instance, live video has stringent timeliness requirements but can tolerate some quality degradation, while radar images must be transmitted reliably and typically require delivery within several tens of seconds. A common theme is that transmitting a larger quantity of and higher quality data from the UAS sensors to the intended recipients (e.g., located at a ground station) is typically desirable.

UAS networks can present **challenges** to transmit sensor data in real-time due to flight dynamics and bandwidth limitations [4], [5], [16]–[19]. However, they also offer **opportunities** since UAS network performance depends on the flight path. To motivate this, we leverage our observations and analysis from two real-world UAS flight test datasets from two different locations in the US (discussed in (§2). Recall that the datasets are unique in that they are from distances exceeding current Federal Aviation Administration (FAA) limits requiring VLOS [20], motivated by the growing interest in extending the range of UAS networking [1], [10], [11].

In this Chapter, we are primarily motivated by the question: is it feasible to **exploit knowledge of UAS flight paths** to more effectively transmit UAS sensor data that has demanding performance requirements? This problem presents multiple key design challenges in dealing with: (i) UAS flight throughput dynamics, (ii) variable sensor data rates and requirements, and (iii) the impact that current sensor data transmission actions have on future transmissions. To address these challenges, we develop **Chimera**, a system for optimizing transmission of heterogeneous sensor data over variable UAS network environments. Chimera is based on an optimal control framework, performing online optimization continuously in order to yield a feedback control policy that makes transmission decisions for the two different sensor data streams: (i) video, and (ii) Synthetic Aperture Radar (SAR) images. While we focus on these streams for concreteness, Chimera can be generalized to more diverse sensors and data streams as well.

A novel aspect of Chimera is its use of flight path information to predict network throughput, and using these predictions in its control algorithm. Towards this end, we develop and validate UAS network throughput prediction models, given flight path knowledge from real-world flights. A key consideration is developing a pragmatic model whose parameters can be learnt online using information from the initial stages of the flight. Our analysis indicates that simple regression models based on the distance and UAS orientation relative to the Ground Control Station (GCS) are effective in prediction, even into the future. We integrate our models into Chimera, which learns both the dependence of throughput on flight path, and also an error model pertaining to throughput prediction errors. Chimera's approach is viable since flight paths are typically determined in advance as part of the mission planning process, and to facilitate flight approval and coordination with proper authorities. Further, Chimera's optimal control framework uses a continual planning model, which allows it to adapt to flight path changes, in addition to learning and improving throughput and error models over time (§4.3.2).

We implemented Chimera and integrated it with a video encoder to stream live video, and an application for generating and transmitting SAR data. We evaluated Chimera using an emulation test-bed, and also built a simulation environment to test a multitude of additional scenarios and evaluate Chimera's design-points. Through a combination of real-world UAS flight throughput traces collected in different locations with different UAS types, and synthetic traces generated using our models, we show that Chimera performs effectively when transmitting diverse sensor data. Chimera offers significant improvement over an approach that does not exploit flight path information. Chimera accomplishes this by reducing penalties related to dropped SAR image transmissions by $72.4\% - 100\%$ across all the aforementioned real-world flight test traces, while achieving comparable video qualities in our

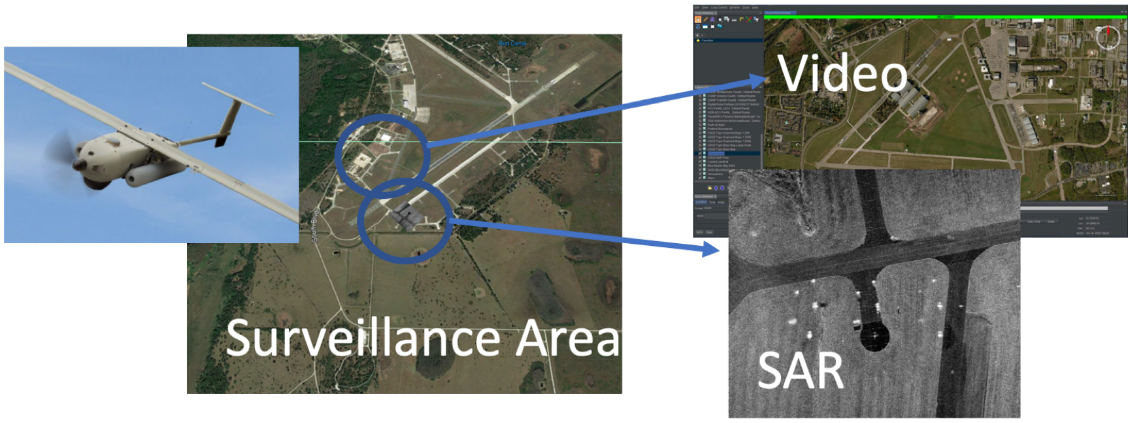


Figure 4.1. Example UAS Surveillance Scenario with SAR + Video

emulation test-bed. Further, Chimera achieves a bitrate of 90.5% compared to an optimal scheme that knows exact future throughput information, with only a modest increase in SAR images dropped. Finally, we believe our models for generating synthetic traces can be useful to the community in their own right.

4.2 Motivating measurements and challenges

In this section, we describe our problem setting for Chimera, and explore the network throughput measurements from real-world UAS flight tests (described in detail in §2) that motivate our approach.

4.2.1 UAS sensor data transmission problem

Many UAS settings, such as security surveillance [7], [24], search and rescue missions [14], [22], and environmental monitoring [23], [25], involve both video and other sensors such as radar imaging [7], [22], [23], called SAR, as shown in Fig. 4.1. Video provides color imaging of small visible areas, while SAR imagery provides a wide-area all-weather capability that penetrates fog, smoke and atmospheric obstructions [23], [26]–[28] (e.g., the ability to penetrate through smoke is critical in fire monitoring scenarios [8], [9]). Live video is often monitored by a person, while SAR images are typically used by algorithms to detect objects and movement. Live video transmission is near real-time and can tolerate loss. In contrast, SAR images are more loss sensitive, but can tolerate greater delays since they cover much wider regions [26], [28]. However, extensive delays (e.g., > 1 minute) can make the images stale and the area may need to be surveyed again. The resolution and priority of each sensor varies based on the mission.

A key challenge that we address in this paper is how to simultaneously transmit both live video and SAR data in challenging UAS networking environments, while taking the requirements of each data stream into account. While we focus on live video and SAR, our approach is easily generalized to diverse sensor data with different reliability and timeliness requirements.

4.2.2 Motivating flight test measurements

Recall from §2 that most existing measurement studies of UAS were performed at limited flight range from the GCS (less than 0.25 miles) or using LTE, which limits scenarios because of required infrastructure [15], [17], [29], [30], [33], [41], [74]. We use the UAS flight test data described in §2, collected in CA and FL and providing flight tests at distances beyond VLOS. For clarity, in this Chapter we use the term *distance* as the absolute distance from the UAS to the GCS.

• *CA Dataset.* Recall that this dataset was collected in CA using a multirotor UAS, omnidirectional antennas on the ground and UAS, and point-to-point Tactical Radios. This data was collected using UDP transport protocol. The abundant throughput from the GCS location **CA0** (see Fig. 2.4), as described in §2, is sufficient such that existing transmission algorithms can work quite well as-is, or with very little modification required. Further, the limited throughput and extreme data loss from the GCS location **CA3** presents challenges that even the most sophisticated algorithms cannot overcome in missions that require sending multiple types of heterogeneous sensor data with reasonable requirements. Thus, we focus our analysis in the rest of this chapter on data collected from the GCS locations **CA1-2**, ranging from 1.8 to 3.4 miles distance from the GCS. The measurements from these locations are shown in Fig. 2.15 and Fig. 2.16, respectively.

• *FL Dataset.* Recall that this dataset was collected in FL using a fixed wing UAS, an omnidirectional antenna on the UAS, and a mixture of both omnidirectional and directional antennas (tested separately) at the GCS. The UAS flight was flown continuously at distances exceeding FAA UAS limits [20], with necessary approvals. We use the directional antenna GCS configuration in Chimera, most appropriate for missions requiring data from multiple sensors that have strict requirements, with the flight path (Fig. 2.3) spanning a distance of up to 7.5 miles between the UAS and GCS. The UAS flew circular patterns at each mile interval, with 0.5 mile radius each, and collected data using two different power levels (500mW and 2W), with performance at the 2W level generally being better at the same range. We refer to these datasets as **FL1** and **FL2**, respectively.

Observations from flight test data: For each dataset, we analyzed the network performance as a function of the UAS distance and orientation in §2. We saw a decrease in throughput as distance increased, and we noticed the performance depended on whether the UAS flies *towards* or *away* from the GCS. The FL throughput is higher than CA at similar distances because the FL dataset used a directional antenna on the ground with higher gain than the CA omnidirectional ground antenna.

Since the FL dataset was collected in one flight from the same GCS, we stitch the circles together in the rest of the chapter to form an extended flight path from 0.5 to 7.5 miles, used for our analysis. We do not stitch the CA dataset together since that was collected with various flights at different GCS locations. Table 4.1 shows our test datasets.

Table 4.1. Flight test traces for Chimera

Name	UAS Type	Distance (Mi)	Power	Antenna
FL1	Fixed Wing	0.5-7.5	500mW	Dir
FL2	Fixed Wing	0.5-7.5	2W	Dir
CA1	Multirotor	1.8-2.6	2W	Omni
CA2	Multirotor	2.6-3.4	2W	Omni

Opportunities and challenges: Overall, our datasets show the dependence of UAS network performance on both distance and UAS orientation, providing an opportunity to plan and optimize data transmission based on the UAS flight path. The key challenges are building throughput prediction models that **exploit knowledge of the UAS flight path**, and **handling throughput prediction errors**. We will tackle these challenges in the rest of this chapter.

4.3 Chimera design

Influenced by the challenges, insights, and opportunities from the relationship of the UAS flight path to network performance, we next outline the problem we are seeking to solve, and the details of our design of Chimera.

4.3.1 Problem formulation

We now formally state our UAS sensor data transmission problem. Consider a flight of duration T seconds. The UAS transmits live video and SAR data, with an associated reward for each. Our objective is to maximize the reward over the full flight. The SAR data stream involves a sequence of images, each B_s bytes, generated every G seconds (typically $<$ 10 seconds) [24], [26], [28], and which must be transmitted within a time limit of L seconds (usually on the order of tens of seconds to a few minutes based on the scenarios described in §4.2). Any image that is transmitted within the deadline L is valuable, and an image that misses the deadline is stale and loses its value. The video transmission objective is to achieve the highest bitrate possible (up to a maximum bitrate needed by that video). We design Chimera to capture the relative importance of video and SAR transmission in a reward function, which we describe in this section. This function rewards higher video bitrates and penalizes dropped SAR images. We often do not have sufficient throughput to transmit the entirety of our data throughout the flight. As a result, Chimera must make decisions on what data to transmit, or delay, at any given time. If we assign too much throughput to video, then SAR cannot keep up with timeliness requirements and becomes stale, resulting in dropped image penalties. If we assign too little throughput to video, then we are potentially wasting opportunity where we could have transmitted higher quality video and still met our SAR requirements.

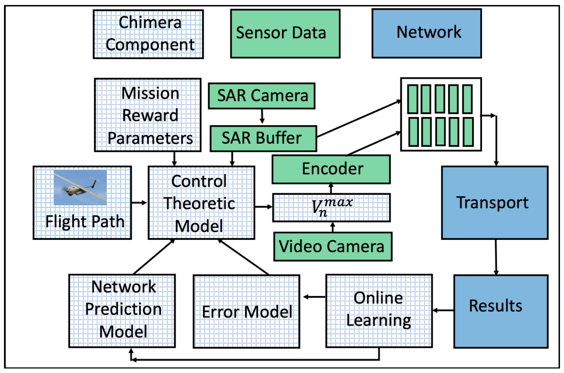


Figure 4.2. Chimera system

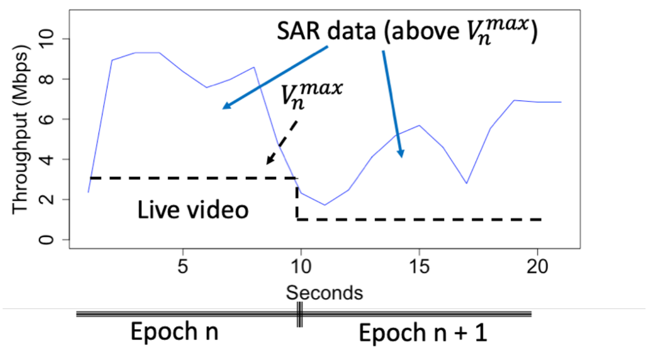


Figure 4.3. Chimera transmission example

4.3.2 Chimera overview

We briefly describe Chimera's four major components and then dive into further details in the subsequent sections (Fig. 4.2 shows a high-level overview).

Control theoretic model: Chimera uses an optimal control framework that yields a feedback control policy that makes transmission decisions at each time-step as a function of current state and predictions of future throughput. In particular, Chimera optimizes expected rewards over the duration of the flight while accounting for prediction errors, which we characterize and model online.

Network prediction model: Chimera's prediction model is built using real-world flight data and utilizes the known UAS flight path to predict future network performance. It includes parameters unique to UAS flight, such as distance and orientation, building a robust model for future throughput prediction. Chimera can work with any generic transport protocol, and uses the network bandwidth estimates provided by the transport layer, with the prediction models trained accordingly.

Adding robustness to prediction errors: Chimera adjusts to network prediction errors by carefully building a weighted probabilistic error model. This error model is integrated into the planning and decision-making process in order to provide robustness to errors and improve performance.

Online learning: It is difficult to gather UAS wireless performance data [5], [17], and flight environments can vary [6]. Because of this, Chimera incorporates an online learning

Table 4.2. Chimera problem notation table

Parameter and notation definitions	
Term	**Meaning**
T, N, τ	flight duration (seconds), flight duration (epochs), epoch duration (seconds)
R, R_d, R_v	total reward, dropped SAR reward penalty, and video reward functions
G, L, B_s	SAR image gen time (seconds), transmission deadline (seconds), size (bytes)
$V_t, \bar{V}_n$	average bitrate for live video during time t, during epoch n
$C_t, \bar{C}_n$	average available throughput during time t, during epoch n
B_{max}, $\bar{b}_n$	max images in buffer, buffer images (fraction) at the start of epoch n
$\bar{G}_n, \bar{D}_n$	SAR images generated during epoch n, dropped at the end of epoch n
$\bar{r}_n, \bar{P}_n$	residual images (fractional) at start of epoch n, transmitted during epoch n
e(n)	subset of all timesteps t representing all timesteps t in the nth epoch
$\bar{\sigma}_n, \bar{\omega}_n, \bar{\epsilon}_n$	distance, orientation, and throughput prediction error at epoch n

process to train network and error models in flight. This online learning process can be used on its own, or be combined with models learnt from previous flights to increase accuracy.

Chimera's approach is viable given that flight paths are typically known *a priori* as part of the mission planning process, and to facilitate flight approval and coordination with proper authorities. However, if flight paths change (e.g., because of an emergency or abrupt mission change), Chimera's feedback control policy enables it to quickly adapt. Specifically, in the case of an abrupt change, Chimera's online optimization would be based on the current system state and network and error models, with consideration of the new flight path. The online optimization would then provide a new set of optimal sensor data transmission decisions for each time-step into the future, based on the new flight path.

Roadmap: In the following sections, (i) we detail Chimera's optimal control model with knowledge of the future throughput (§4.3.3), (ii) we develop models for predicting throughput using real-world flight datasets (§4.3.4), and (iii) we discuss how Chimera builds its throughput and error models online, and Chimera's algorithmic approach that utilizes them (§4.3.5).

4.3.3 Control theoretic formulation

Let there be T second time-steps, indexed by $t \in \{0, 1, 2, \ldots, T-1\}$. The throughput at each time-step is C_t. Our goal is to allocate the throughput between live video and SAR traffic at each time-step so as to optimize the overall reward.

As we have discussed, a unique opportunity in UAS settings is that the entire flight path is typically planned in advance, providing the potential to use a long term look-ahead window for planning. For computational efficiency reasons (as we will discuss further in §4.5), it may sometimes be desirable to conduct Chimera's planning at a coarser granularity than at each time-step. Towards this end, we partition the duration of the flight into N epochs, each of a fixed length of time τ. We index the epochs by the variable $n \in \{0, 1, 2, \ldots, N-1\}$. We next present Chimera's control model (Table 4.2 summarizes notation).

Video: Let V_t be the achieved live video bitrate at a given time-step t (such that $V_t <= C_t$). Consider a sequence of achieved live video bitrates at each time-step: $V_0, V_1, \ldots, V_{T-1}$. The achieved video bitrate over the entire flight is $\sum_{t=0}^{T-1} V_t$, and we can divide by T to obtain the average transmitted video bitrate per second over the entire flight. To enable epoch level decision-making, we define the quantity $\mathrm{e}(n)$, where for each $n \in \{0, 1, \ldots, N-1\}$, $\mathrm{e}(n)$ is the set of all time-steps t in the nth epoch. Thus, our average live video bitrate for epoch n, denoted $\bar{V}_n$ is:

$$\bar{V}_n = \frac{\sum_{t \in \mathrm{e}(n)} V_t}{\tau}. \tag{4.1}$$

We select video values in order to maximize the reward (provided later in Eq. (4.8)) by influencing the balance of dedicated throughput to live video and SAR. To do this, we specify the maximum live video bitrate transmitted within an epoch, denoted $\bar{V}_n^{max}$, selected from a finite set of possible bitrates $\mathcal{V}$. The encoder targets (and never exceeds) a bitrate of $\bar{V}_n^{max}$ in each epoch, but adapts to dips in network throughput by encoding at lower bitrates as needed. For each $t \in \mathrm{e}(n)$, we let $V_t \in [0, \bar{V}_n^{max}]$ denote the realized throughput of live video during time-step t:

$$V_t = \min(C_t, \bar{V}_n^{max}). \tag{4.2}$$

Note that V_t will be a random variable, determined by actual throughput C_t during that time-step. The remaining throughput $C_t - V_t$ in any given time-step is dedicated to SAR transmission, through a scheduler. Fig. 4.3 shows an example of Chimera over a period of 20 seconds, split into 10 second epochs. During each epoch n, $\bar{V}_n^{max}$ is a dotted line and excess throughput during that epoch is devoted to SAR transmission. For each epoch, we seek a $\bar{V}_n^{max}$ that is achievable but also leaves sufficient throughput for SAR transmission. Without this limitation, SAR and video flows would compete with each other, and potentially starve the system. By predicting future throughput based on the UAS flight path, and varying data prioritization at different points during flight, we can improve our reward.

SAR image transmission: After allocation of throughput to video, the remaining throughput for each epoch is dedicated to SAR images, given by $\sum_{t \in \mathrm{e}(n)}(C_t - V_t)$ (note that remaining throughput for SAR can be zero if $\bar{V}_n^{max}$ is equal to or exceeds the available throughput at each time-step during epoch n). This strategy allows for long-term planning to maximize reward, while also adapting live video to short term fluctuations in throughput. The use of multiple levels of planning and adaptation takes advantage of long-term horizon throughput prediction based on flight path and also allows faster computations compared to second-by-second planning and processing.

SAR buffer: During each epoch n, $\bar{G}_n$ new full (integer) SAR images are generated and stored in the buffer for transmission. Let $\bar{b}_n$ denote the number of SAR images (fractional) stored in the buffer at the start of epoch n, and let $\bar{P}_n$ denote the number of SAR images (fractional as partial images can be in transmit) transmitted to the GCS during that epoch:

$$\bar{P}_n = \min\left\{\frac{\sum_{t \in \mathrm{e}(n)}(C_t - V_t)}{B_s}, \bar{b}_n\right\}. \tag{4.3}$$

An incomplete SAR image is not valuable to us. We must transmit the entire image to provide useful information to the system operator and get the reward for that image. Let

$\bar{r}_n$ be our residual (partial) images and $\bar{S}_n$ be the number of full SAR images (integer) transmitted during epoch n, given by:

$$\begin{aligned}
\bar{r}_n &= \bar{b}_n - \lfloor \bar{b}_n \rfloor, \\
\bar{S}_n &= \begin{cases} 0 & \text{if } \bar{r}_n > \bar{P}_n \\ \lfloor \bar{P}_n \rfloor & \text{if } \bar{r}_n = 0 \\ 1 + \left\lfloor \bar{P}_n - \bar{r}_n \right\rfloor & \text{else} \end{cases}.
\end{aligned} \tag{4.4}$$

Let B_{max} denote the maximum number of full SAR images that can be stored in the buffer (calculated based on the deadline, L, to transmit SAR images before they become stale). L is a multiple of the generation time, G, such that:

$$B_{max} = L/G. \tag{4.5}$$

Let $\bar{D}_n$ denote the number of SAR images dropped from the buffer during epoch n (penalized as an integer, even if part of the image was transmitted because only full SAR images transmitted are useful to us). The dynamics of the buffer are:

$$\begin{aligned}
\bar{b}_0 &= 0, \\
\bar{b}_{n+1} &= \min\{\bar{b}_n - \bar{P}_n + \bar{G}_n, B_{max}\}, n \in \{0, 1, \ldots, N-1\}.
\end{aligned} \tag{4.6}$$

Furthermore, the number of SAR images dropped from the buffer at the end of each epoch n is given by:

$$\bar{D}_n = \max\{0, \lceil \bar{b}_n - \bar{P}_n + \bar{G}_n - B_{max} \rceil\}. \tag{4.7}$$

Decisions and rewards: Consider a sequence of specified maximum live video bitrates:

$$\bar{V}^{max}_{0:N-1} \triangleq \{\bar{V}^{max}_0, \bar{V}^{max}_1, \ldots, \bar{V}^{max}_{N-1}\} \in \mathcal{V}^N,$$

where each $\bar{V}^{max}_n$ denotes the maximum live video bitrate during epoch n. Each such sequence induces a sequence of achieved live video bitrates and dropped SAR images. Note that each of these quantities is a random variable, dependent on the realized throughput during each epoch. During flight, the system operator obtains a live video transmission reward, $R_v(V_t)$, every second. The system operator also incurs a penalty, R_d, for the total SAR images dropped during the flight, which can be represented as the summation of images dropped at each time-step, or equivalently the summation of images dropped across epochs.

Flight reward with epochs: The *expected reward* earned over the duration of the flight for a given sequence $\bar{V}^{max}_{0:N-1}$ is:

$$R(\bar{V}^{max}_{0:N-1}) = \mathbb{E}\left[\sum_{t=0}^{T-1} R_v(V_t) - R_d(\sum_{n=0}^{N-1} \bar{D}_n)\right]. \tag{4.8}$$

Optimization problem with epochs: Our optimization problem based on epochs is:

$$\begin{aligned} &\max_{\bar{V}^{max}_{0:N-1} \in \bar{\mathcal{V}}^N} R(\bar{V}^{max}_{0:N-1}) \\ &\text{subject to (4.2), (4.3), (4.5), (4.6), and (4.7).} \end{aligned} \tag{4.9}$$

Chimera extensions: Chimera is extendable to different reward functions using the above framework. For example, it is easy to add a reward for successful full SAR image transmissions, $\bar{S}_n$ in Eq. (4.4), instead of, or in addition to penalties for dropped SAR images. It is also possible to extend the reward function to minimize fluctuations in the live video bitrate by adding a smoothness function, R_Δ, that penalizes changing live video bitrates. To do this, we would calculate the difference in video bitrates at each time-step: $\Delta_t = V_{t+1} - V_t$. Each change in video bitrate would result in a corresponding penalty based on the smoothness function, $R_\Delta(\Delta_t)$, and a summation of these penalties may be taken.

4.3.4 Throughput prediction model with Chimera

Our discussions in §4.3.3 assume we know the throughput for each time-step, C_t, in advance. In this section, we develop and validate models to capture how network throughput depends on the UAS flight path, leveraging the real-world flight datasets discussed in §4.2. Our models consider three key factors: (i) UAS distance to the GCS, (ii) UAS orientation, and (iii) recent throughput samples. We first look at prediction for the immediate next time-step, and then consider longer look-aheads.

Prediction over the next time-step: Our data shows that UAS distance to the GCS affects throughput. We built a regression model to calculate predicted throughput at time t, $\hat{C}_t$, based on distance, σ_t, with an estimated error term, $\hat{\epsilon}_t$:

$$\hat{C}_t = \alpha\sigma_t + \gamma + \hat{\epsilon}_t. \tag{4.10}$$

In Eq. (4.10), the coefficients α *and* γ are calculated based on gathered data. We considered a distance to throughput relationship of linear, log, and a combination of both. We built a regression model for each relationship and found the linear relationship to be slightly more accurate than log, and the same as the combination (setting the log coefficient to 0).

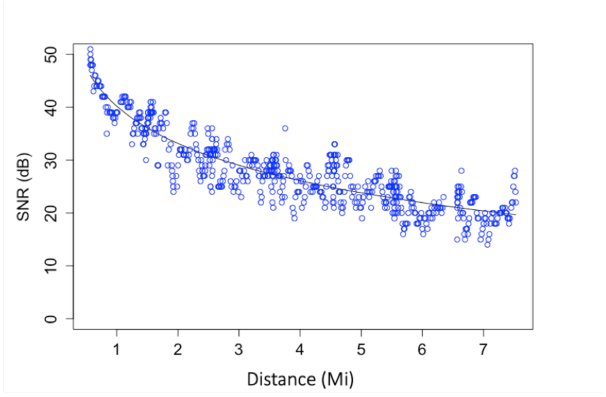


Figure 4.4. SNR-dist relationship

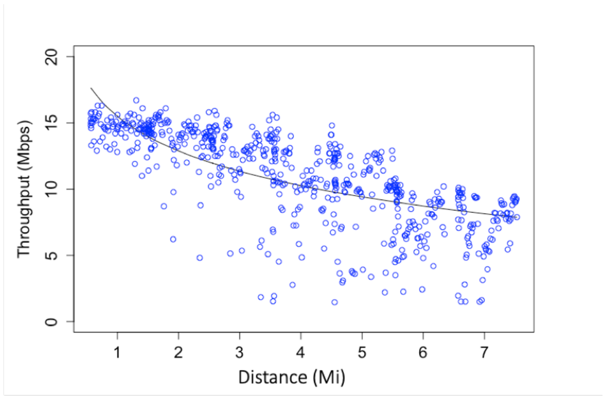


Figure 4.5. Throughput-dist relationship

We also explored the relationship of Signal-to-Noise Ratio (SNR) to distance. SNR, measured in dB, is expected to have a logarithmic relationship to distance [75]–[77]. Fig. 4.4 verifies this for our data and shows that the SNR exhibits a relationship with distance that follows a logarithmic curve. However, Fig. 4.5 shows that throughput and distance do not clearly exhibit a logarithmic relationship, and visually appear to have a relationship that is more along the lines of linear, in agreement with our regression model results. We hypothesize that this is because of the many layers involved in network communication, and SNR is only one of the contributing factors that impact throughput.

Considering orientation: We built two regression models: one using the entire dataset (*orientation agnostic*) and the other with separate models for each orientation. We then compared the throughput prediction of these models to the actual throughput across the entire flight. Fig. 4.6 (left) shows prediction errors are reduced by building seperate models for **CA1** and **FL1**. The other datasets were qualitatively similar.

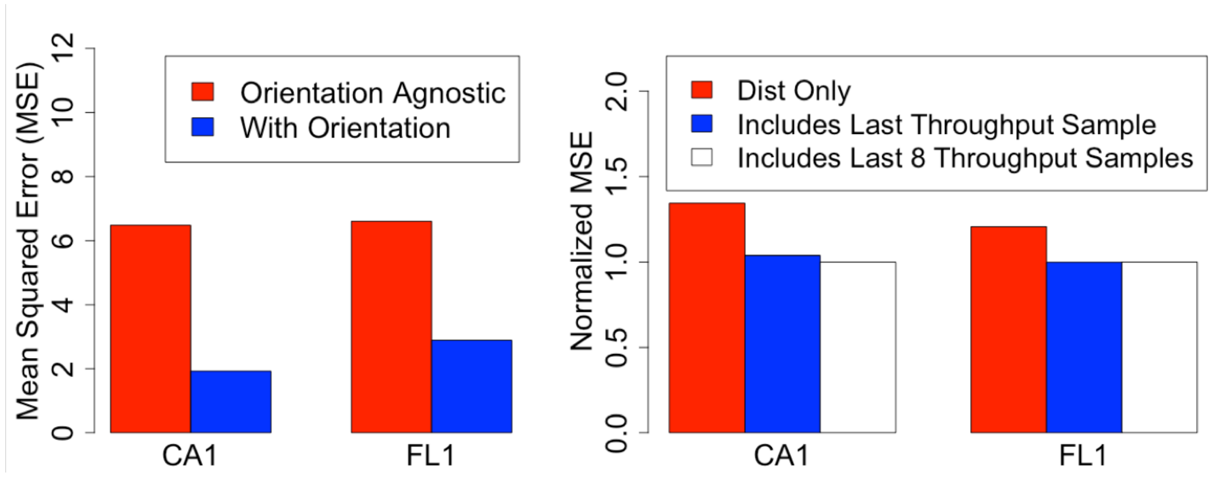


Figure 4.6. Throughput prediction model differences with orientation (left) and inclusion of previous throughput samples (right), normalized to the last 8 throughput samples.

Previous throughput samples: We explore if we can improve our model by including additional previous throughput samples. We modify Eq. (4.10) to include previous throughput samples and calculate regression models using 0, 1, and 8 previous throughput samples. We then use these regression models to compare predicted throughput to the actual throughput over the same time period. Fig. 4.6 (right) shows that incorporating previous throughput improves prediction accuracy. However, the improvement gained by including throughput samples beyond the previous time-step is small. The other datasets were qualitatively similar.

Network model for Chimera: Based on these results, Chimera uses a regression model for throughput prediction at time t, $\hat{C}_t$, based on C_{t-1} (the actual throughput at time $t-1$), an estimated error term $\hat{\epsilon}_t$, and ω_t and σ_t, which respectively denote the orientation and distance at time t:

$$\hat{C}_t = \begin{cases} \delta C_{t-1} + \alpha\sigma_t + \gamma + \hat{\epsilon}_t & \text{if } \omega_t = 0 \\ \delta' C_{t-1} + \alpha'\sigma_t + \gamma' + \hat{\epsilon}_t & \text{if } \omega_t = 1. \end{cases} \tag{4.11}$$

In Eq. (4.11), the coefficients $\delta,\ \alpha,\ \gamma,\ \delta',\ \alpha',\ and\ \gamma'$ are calculated based on gathered flight data, while ω_t indicates the orientation (i.e., $\omega_t = 0$ if the UAS is **going away** at timestep t and $\omega_t = 1$ if the UAS is **coming towards**).

Prediction over a longer look-ahead: So far, we considered a regression model for predicting throughput at the next time-step. However, for Chimera, we must predict throughput over longer time horizons (in fact, the entire flight path). Consider that throughput samples up to time $t-1$ are available. We seek to predict throughput for time-steps $t, t+1, \ldots, t+k$. Extending our analysis, we consider a model where throughput at time-step $t+k$, C_{t+k}, depends on C_{t-1} (the last throughput sample), ω_{t+k} and σ_{t+k} (the orientation and distance at time $t+k$), and $\hat{\epsilon}_{t+k}$ (the estimated error):

$$\hat{C}_{t+k} = \begin{cases} \delta_k C_{t-1} + \alpha_k\sigma_{t+k} + \gamma_k + \hat{\epsilon}_{t+k} & \text{if } \omega_t = 0 \\ \delta'_k C_{t-1} + \alpha'_k\sigma_{t+k} + \gamma'_k + \hat{\epsilon}_{t+k} & \text{if } \omega_t = 1. \end{cases} \tag{4.12}$$

Significance as time lag (k) increases: Fig. 4.7 compares throughput prediction errors for **FL1**, using Eq. (4.12), with different coefficients turned off (i.e., considering distance + past throughput, and each on its own). We repeat this regression for each lag k, iterating through all t values for each k value in our flight, and show the MSE across lags. The left side considers the **going away** orientation and the right considers **coming towards**. The results show that (i) distance is important, and (ii) only considering previous throughput sample is insufficient. Further, the previous throughput sample helps for low lag, but not larger lag. We also see a larger error in the **coming towards** orientation compared to **going away**, owing to higher variability based on orientation, as discussed in §2. We see a slight oscillation in MSE when considering past throughput only, but it is reduced when distance is considered, since the distance coefficient magnitude is more significant. Based on these results, we remove the previous throughput samples from our long-term prediction model.

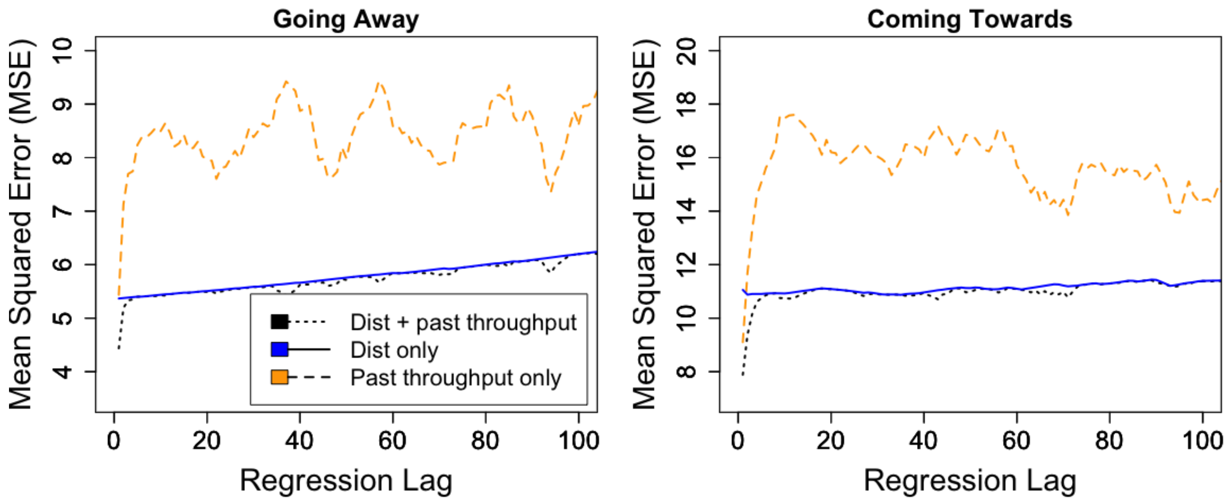


Figure 4.7. MSE across different time lags when considering distance + past throughput, or each individually, **FL1**.

Prediction over epochs Chimera may make decisions at a coarser granularity of epochs, as discussed in §4.3.3. This is done by using epochs instead of time-steps in Eq. (4.11), where the epoch distance and throughput ($\bar{\sigma}_n$ and $\bar{C}_n$) are the average distance and throughput over all time steps within that epoch. Further, we remove dependence on previous throughput, as previously discussed, and our model for throughput prediction over epochs is:

$$\hat{\bar{C}}_n = \begin{cases} \alpha\bar{\sigma}_n + \gamma + \hat{\bar{\epsilon}}_n & \text{if } \bar{\omega}_n = 0 \\ \alpha'\bar{\sigma}_n + \gamma' + \hat{\bar{\epsilon}}_n & \text{if } \bar{\omega}_n = 1. \end{cases} \tag{4.13}$$

We built regression models with Eq. (4.13) using different epoch sizes. Throughput dependence on both UAS distance and orientation continued to hold at the epoch scale. Further, the average throughput prediction error within an epoch decreased as epoch size increased, with median prediction errors of 15.3%, 11.8%, 9.0%, and 5.2% for epoch sizes of 1, 5, 10, and 30, respectively. However, there may still be significant throughput variability within an epoch, and larger epochs may reduce Chimera's responsiveness (§4.5.3).

4.3.5 Incorporating predictions into Chimera

We next discuss how we integrate the network models in §4.3.4 into Chimera's planning algorithm, run each epoch.

Online optimization: We implement Chimera's online optimization as a Dynamic Program (DP) that incorporates the UAS flight network characteristics to maximize reward (with evaluation results shown in §4.5). The full DP, with throughput prediction, error models, and probabilistic errors, is shown in Algorithm 1. We start by discussing the basic DP and then discuss how an error model is integrated. The state for the DP is the number of untransmitted SAR images (the SAR buffer, b_n). For each epoch and state, the DP computes the optimal sequence of targeted (and maximum) live video bitrate values, $\bar{V}_n^{max}$, starting at the next epoch until the end of the look-ahead window (by default, the end of the trace), maximizing the total reward.

Let the throughput prediction for the next epoch be $\hat{\hat{C}}_n$. For each value of $\bar{V}_n^{max}$, the average live video bitrate transmitted during that epoch, $\bar{V}_n$, is limited to $\min\{\hat{\hat{C}}_n, \bar{V}_n^{max}\}$ because the video bitrate cannot exceed the available throughput. We modify Eq. (4.3) to determine the (possibly fractional) SAR images transmitted during an epoch, $\bar{P}_n$, as follows:

$$\bar{P}_n = \min\{\tau \frac{\hat{\hat{C}}_n - \min\{\hat{\hat{C}}_n, \bar{V}_n^{max}\}}{B_s}, \bar{b}_n\}. \tag{4.14}$$

Note that if $\hat{\hat{C}}_n \leq \bar{V}_n^{max}$ then $\bar{P}_n = 0$. In this case, our throughput estimate is such that the target live video bitrate cannot be achieved. Available throughput will be devoted to live video transmission, estimated as $\hat{\hat{C}}_n$, and no SAR data will be transmitted. The rest of the computations follow §4.3.3. Our test results showing the benefits of Chimera are presented in §4.5.

Incorporating prediction error: To be robust to prediction inaccuracies, we build an error model online. Specifically, Chimera maintains a distribution of errors in predictions made earlier in the flight. Next, Chimera selects a few discrete points in the distribution (in our implementation, we select quartiles). Given a predicted throughput, $\hat{\hat{C}}_n$, Chimera considers each quartile error, ϵ_q, adjusting the throughput prediction for each error as follows:

Algorithm 1: Dynamic Program

```
Result: S is our optimal decision for video with maximum reward R at each epoch
        and buffer size. Starting at R[0,0] for epoch and buffer 0.
Initialize;
for $n = (N-1)$ to $0$ do
    for $b = 0$ to $b_{max}$ do
        $R_{max} = -\infty$;
        for $V_n^{max} \in V$ do
            $R_{prob} = 0$;
            for $k$ = 0 to binSize do
                $\tilde{\epsilon}_k = probError\ at\ k$;
                $\tilde{Pr}_k = Pr[\tilde{\epsilon}_k]$;
                $\hat{\bar{C}}_k = \hat{\bar{C}}_n + \hat{\bar{C}}_n \times \tilde{\epsilon}_k$;
                if $\hat{\bar{C}}_k > \bar{V}_n^{max}$ then
                    $\bar{P}_n = \min\{\tau \frac{\hat{\bar{C}}_k - \bar{V}_n^{max}}{B_s}, b\}$;
                    $\bar{V}_n = \bar{V}_n^{max}$;
                    $\bar{r}_n = b - \lfloor b \rfloor$;
                    if $\bar{r}_n > \bar{P}_n$ then
                        $\bar{S}_n = 0$;
                    else
                        if $\bar{r}_n = 0$ then
                            $\bar{S}_n = \lfloor \bar{P}_n \rfloor$;
                        else
                            $\bar{S}_n = 1 + \lfloor \bar{P}_n - \bar{r}_n \rfloor$
                else
                    $\bar{S}_n, \bar{P}_n = 0$;
                    $\bar{V}_n = \hat{\bar{C}}_k$;
                $b' = b - \bar{P}_n + \bar{G}_n$;
                $\bar{D}_n = \max\{0, \lceil b' - \bar{B}_{max} \rceil\}$;
                $\tilde{R}_k = \tau R_v(\bar{V}_n) - R_d(\bar{D}_n) + R[n+1, discretize(b')]$;
                $R_{prob} = R_{prob} + \tilde{R}_k \times \tilde{Pr}_k$;
            if $R_{prob} > R_{max}$ then
                $R_{max} = R_{prob}$;
                $S[n,b] = \bar{V}_n^{max}$;
                $R[n,b] = R_{prob}$;
```

$\hat{\hat{C}}_q = \hat{\hat{C}}_n + \hat{\hat{C}}_n \times \epsilon_q$. Our DP evaluates the choice of $\bar{V}_n^{max}$ with each error-adjusted throughput prediction, weighing the resulting reward according to the error probability. Chimera considers the summation of these weighted rewards to select the optimal choice of $\bar{V}_n^{max}$ when evaluating each epoch. Our results show that incorporating an error model improves Chimera's performance (§4.5.3).

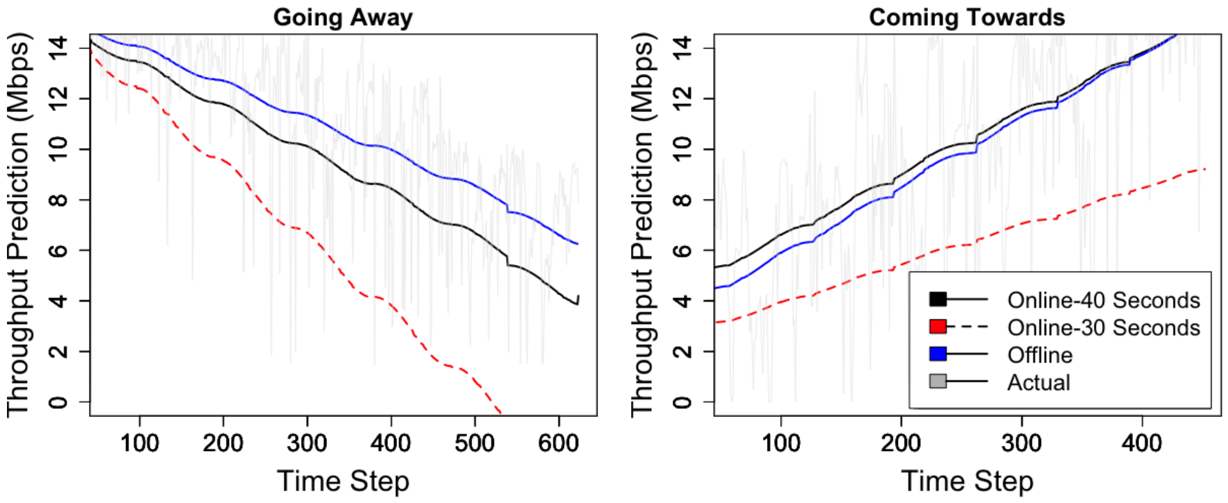


Figure 4.8. Comparison of online learning throughput prediction, after 30 and 40 seconds, compared to offline learning.

Online learning: Chimera uses online learning for generating throughput prediction and error models. At the start of each epoch, Chimera calculates the previous throughput prediction error and reruns the throughput regression model in §4.3.4 with all observed flight throughput data, allowing Chimera to adjust to changes and update predictions during flight. Figure 4.8 illustrates Chimera's online learning with the **FL1** trace, and shows the throughput predicted by the regression models learnt online after 30 and 40 seconds, compared to the prediction from an offline regression model that has all of the data ahead of time. The figure shows that Chimera can converge to the offline model within tens of seconds, a trend we also observed in other traces. This furthers our confidence in Chimera being able to quickly adapt to different environments.

An important consideration is how much previous data should be included in online learning. We experimented with several strategies for learning throughput and error models. For throughput, we considered (i) building a single regression model for the entire flight using all of the available data up to that point in the flight and (ii) building separate regression models based on orientation. For error models, we considered both strategies of using all of the available flight data verses building separate models based on orientation. Further, we also considered building separate error models based on distance ranges (e.g., distance bins covering 2 miles each) and orientation, motivated by the throughput prediction error distribution in §4.4. We show in §4.5 that building orientation-specific models typically improves performance for both error and throughput models, with the most benefits in traces that encounter each orientation more than once during flight. Further, we found the addition of error models learnt online based on distance bins was similar to and did not improve Chimera's performance beyond using orientation-specific models-potentially due to limiting the data available for learning. Finally, while we focus on online learning for the most general settings, we note that UAS settings may have error and/or throughput model data from prior flights that Chimera can also leverage (shown in §4.5.3).

Chimera Oracle optimization algorithm: Algorithm 2 shows Chimera's DP with a perfect Oracle look-ahead. In this variant, the equations follow §4.3.3 exactly as there is no need for error models, since the throughput for the entire flight is known by the perfect Oracle. Thus, the actual throughput at each time-step, C_t, is used in the DP calculations, rather than a predicted average throughput for each epoch, as is done in Algorithm 1. The result is that detailed planning and rewards are calculated at the time-step level for live video bitrate and actual throughput, since the Oracle provides this granularity of perfect throughput knowledge. This version of the DP is used to provide an example of the optimal performance for each trace based on exact knowledge of future throughput (while not possible in practice, it presents a best-case scenario for comparison).

4.4 Evaluation methodology

We evaluate Chimera with an actual implementation on an emulation test-bed and using simulations, as we discuss below.

Implementation and test-bed Setup: We implemented Chimera and integrated it with a VP8 video encoder to stream live video, and an application for generating and trans-

Algorithm 2: Dynamic Program (Oracle)

```
Result: S is our optimal decision for video with maximum reward R at each epoch
        and buffer size. Starting at R[0,0] for epoch and buffer 0.
Initialize;
for $n = (N-1)$ to $0$ do
    for $b = 0$ to $b_{max}$ do
        $R_{max} = -\infty$;
        for $V_n^{max} \in V$ do
            $V_t = \min\{C_t, V_n^{max}\} \ \forall \ t \in \mathrm{e}(n)$;
            $\bar{P}_n = \min\{\frac{\sum_{t\in \mathrm{e}(n)}(C_t - V_t)}{B_s}, b\}$;
            $\bar{r}_n = b - \lfloor b \rfloor$;
            if $\bar{r}_n > \bar{P}_n$ then
                $\bar{S}_n = 0$;
            else
                if $\bar{r}_n = 0$ then
                    $\bar{S}_n = \lfloor \bar{P}_n \rfloor$;
                else
                    $\bar{S}_n = 1 + \lfloor \bar{P}_n - \bar{r}_n \rfloor$
            $b' = b - \bar{P}_n + \bar{G}_n$;
            $\bar{D}_n = \max\{0, \lceil b' - \bar{B}_{max} \rceil\}$;
            $R' = \sum_{t\in \mathrm{e}(n)} R_v(V_t) - R_d(\bar{D}_n) + R[n+1, discretize(b')]$;
            if $R' > R_{max}$ then
                $R_{max} = R'$;
                $S[n,b] = \bar{V}_n^{max}$;
                $R[n,b] = R'$;
```

mitting SAR data. We leveraged the codebase from Salsify [78], while making modifications to inform the encoder of a target bitrate by Chimera, as described in §4.3. We used Mahimahi [71] to emulate flight network throughput, replaying the traces from the flight test datasets, described in §4.2. Our control setup integrated both the distance and orientation of the UAS, allowing Chimera to continuously run the online optimization, and make decisions based on the flight path and current state, while also providing an opportunity to adapt to changes in the progressive throughput and error models. We generate representative SAR image files and transmit these to the client GCS at a set rate that is updated every second based on the estimated remaining available throughput, per our protocol design. Our tests are run on a 64-bit Ubuntu 20.04 2-core machine, with 8 GB RAM - representative of a typical UAS system. We also tested at a large scale with a simulated setup that integrated the distance and orientation of the UAS into the logic, in order to provide more extensive sensitivity studies.

Real-world traces: We test using the throughput traces from multiple flights in FL and CA (§4.2, Table 4.1). We also perform tests with dataset variants, which we discuss in §4.5. UAS flights typically involve several to tens of minutes of flight time. Since our CA flight traces are shorter duration, we extended them by an additional 3 loops, by synthetically generating throughput loops (as described below) and appending them to our dataset (this is reasonable because UAS survey missions often complete multiple loops of an area).

Synthetic traces: Since real-world flight traces are challenging to collect, we generated synthetic traces for additional testing using the network model from §4.3.4 (Eq. (4.11)), and coefficients gleaned from our real-world flight data. Since the errors are higher with larger distances and certain orientations (e.g. **coming towards**), we model the error term, ϵ_t, separately for different distance range and orientation bins, using a procedure described below.

Fig. 4.9 shows a histogram of prediction errors, and the best fit for both a Normal and Cauchy distribution, for the FL1 trace and an example bin (a distance range of 2 to 4 miles). Visually, we see the Normal distribution is not a good fit because it does not encompass the peak or tails, while the Cauchy distribution is a better fit. We used the Kolmogorov-Smirnov (KS) test [79] to evaluate the null hypothesis that the prediction errors are drawn

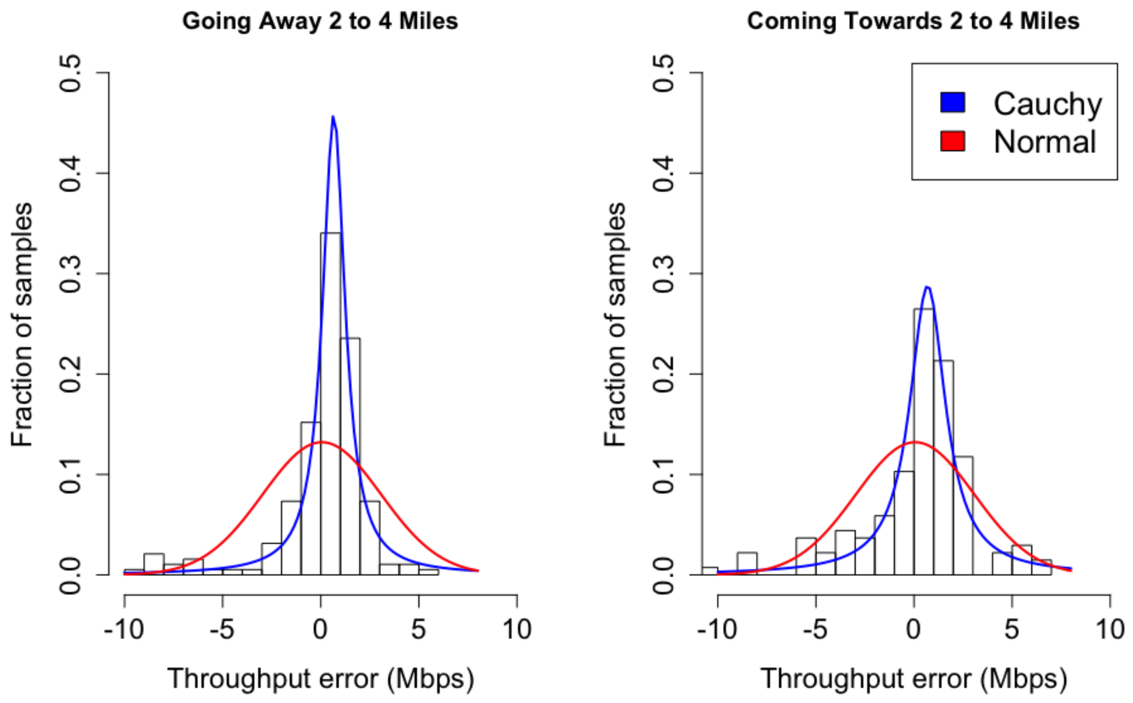


Figure 4.9. Modeling the prediction error term for synthetic trace generation

from a specific distribution. We considered the Normal, Cauchy, Lognormal, Gamma, and Weibull distributions. For all bins of the FL and CA datasets, we were unable to reject the null hypothesis that the prediction errors fit a Cauchy distribution at a significance level of 0.05. In contrast, we rejected the null hypothesis that the errors fit a Lognormal and Gamma distribution in all cases, and in the vast majority of cases for Normal and Weibull distribution (the full KS test results for **FL1** are shown in Table 4.3 and Table 4.4 for each orientation, respectively). We considered both percentage and absolute errors, with consistent results. We tested with the Anderson-Darling (AD) and Cramer Von-Mises (CVM) tests, which are refinements of the KS test [79], with nearly identical results to the KS test, supporting the Cauchy distribution as a good fit for our error distributions.

These results motivated us to model errors based on the Cauchy distribution, confirming what we visually saw in Fig. 4.9: that the error distribution has a higher peak and longer tails than can be captured with a Normal distribution. In summary, we generate synthetic traces using Eq. (4.11), with the error term generated using a Cauchy distribution, and

Table 4.3. Kolmogorov-Smirnov test results (going away error bins), **Florida1**

Distribution	**Under 2 Miles**	**2-4 Miles**	**4-6 Miles**	**Over 6 Miles**
Cauchy	Not Rejected	Not Rejected	Not Rejected	Not Rejected
Normal	Rejected	Rejected	Not Rejected	Not Rejected
Lognormal	Rejected	Rejected	Rejected	Rejected
Gamma	Rejected	Rejected	Rejected	Rejected
Weibull	Rejected	Rejected	Rejected	Not Rejected

Table 4.4. Kolmogorov-Smirnov test results (coming towards error bins), **Florida1**

Distribution	**Under 2 Miles**	**2-4 Miles**	**4-6 Miles**	**Over 6 Miles**
Cauchy	Not Rejected	Not Rejected	Not Rejected	Not Rejected
Normal	Not Rejected	Rejected	Rejected	Rejected
Lognormal	Rejected	Rejected	Rejected	Rejected
Gamma	Rejected	Rejected	Rejected	Rejected
Weibull	Rejected	Rejected	Rejected	Rejected

parameters based on the flight distance and orientation. To ensure meaningful results (e.g., avoid negative throughput), the tail was truncated on both sides.

Schemes: We compared Chimera with several schemes:

• *Flight Agnostic*: Our baseline scheme uses a model-predictive controller with a look-ahead window of 5 epochs, and throughput prediction based on the average throughput in the past 5 epochs. In each look-ahead window, the same DP as Chimera is run to best allocate bandwidth between the live video and SAR data streams. This scheme does not exploit knowledge of the UAS flight path, and is inspired by an algorithm widely used in the context of Internet video streaming [50].

• *Oracle*: This scheme assumes perfect knowledge of throughput for the duration of the trace at the start of the flight, and executes a DP based on the model in §4.3.3.

• *Chimera and variants*: By default, we evaluate *Chimera* with all its features, including online learning and a probabilistic error model. We also explore several variants to test key decisions of Chimera, which we detail later.

Evaluation settings: We consider a reward function (§4.3.3) based on live video and SAR image transmission. We set the reward for live video at each time-step, $R_v(V_t)$, to be equal to the live video bitrate received at that time-step. Let the maximum possible reward for live video transmissions be M (achieved when live video is always transmitted at the highest rate), I be the total number of SAR images generated, and D be the total number of dropped images during flight. Then, we set the penalty for dropped SAR images to be $W \times \frac{M}{I} \times D$. Here, W is a parameter which captures that if all SAR images were dropped, the penalty would be W times the reward obtained if live video were always transmitted at the highest rate. We use a default W of 8 since a dropped SAR image implies the data is completely lost, while live video could still be transmitted at degraded bitrates. We also evaluate Chimera with different W values. We set possible values of $\bar{V}_n^{max}$ to be $\{1, 2.5, 5\}$, corresponding to bitrates typical for standard and high definition video [80]. Our SAR sensor generates a full image every 5 seconds, based on a real-world system [24]. We use SAR images of 16, 32, and 40 Mb for **CA1-2**, **FL1**, and **FL2**, respectively, modeling higher resolution images in datasets with better throughput. We set the SAR transmission deadline to be $L = 60$ *seconds*.

4.5 Results

Our test results show that (i) utilizing UAS flight path improves performance (beneficial to use both distance and orientation), (ii) performance is further increased by using a probabilistic error model, and (iii) online learning performs well, even with no prior knowledge or model insights.

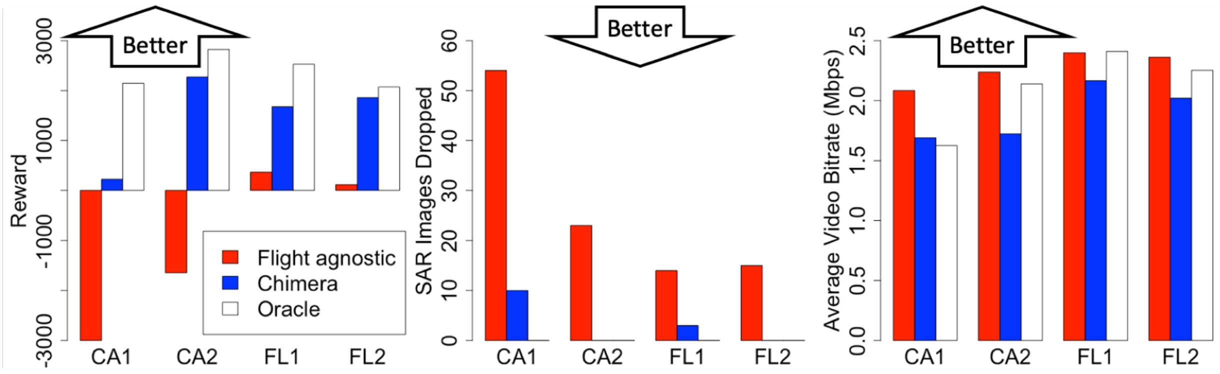


Figure 4.10. Chimera's performance with emulation testing, showing comparisons of reward (left), SAR images dropped (middle), and the average live video bitrate (right).

4.5.1 Effectiveness of Chimera: emulation results

Fig. 4.10 shows a comparison of the Flight Agnostic, Chimera, and Oracle schemes with all of our traces on the emulation testbed. The left-most figure corresponds to the total reward, while the other figures show the performance of each sensor stream: SAR images dropped due to becoming stale (middle), and the average live video bitrate measured at the receiver (right). We see Chimera performing much better than the Flight Agnostic scheme because it significantly reduces SAR drops, while maintaining a comparable video bitrate (the negative Y-Axis is truncated at -3000, and Flight Agnostic achieves a reward of -8049.12 in **CA1**). Flight Agnostic performs worse because it is more aggressive with live video bitrate transmission and does not properly prepare for periods of poor throughput. In contrast, by utilizing knowledge of the flight path, Chimera and the Oracle (with perfect knowledge) account for future periods of lower throughput by throttling back the maximum permitted live video bitrate. Consequently, the SAR buffer (number of untransmitted images) fills up faster for Flight Agnostic with 54 images dropped in (**CA1**), while only 10 images are dropped with Chimera.

While Chimera performs comparably to the Oracle in most traces, there is a noticeable gap to the Oracle for the **CA1** trace. Upon further inspection, this is because **CA1** involved a sharp and prolonged drop in throughput when transitioning to the orientation with poor performance for the first time, leading to dropped images. Chimera starts with no prior knowledge, but its performance improves over time as it quickly learns better throughput and error models. We note that UAS flights are typically longer, which allow online learning approaches to work even better. Finally, Chimera can improve performance using models learnt from previous flights (§4.5.3), which may be available in many scenarios.

4.5.2 Sensitivity to traces

Since we are not aware of any other real-world UAS flight datasets at long-range distances, we test Chimera with several additional traces, described below. Our results are based on simulations. All schemes perform slightly better using the simulator (since it does not account for factors such as processing and encoding delays), but we verified the relative performance is similar to the emulator. Across the traces used for validation, Chimera achieved 63.1% of the reward of the Oracle in emulation, compared to 63.6% of the Oracle reward in simulation.

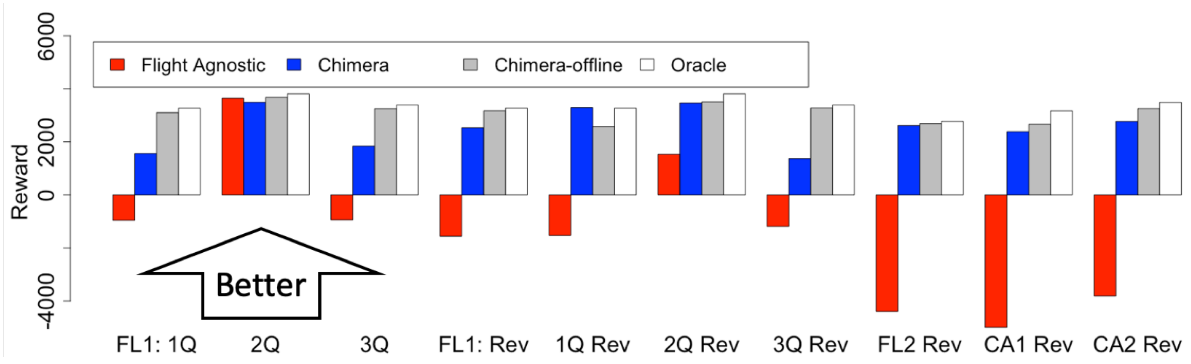


Figure 4.11. Chimera reward performance for variants, showing simulated testing of starting in different positions with **FL1**, and also the reverse direction for all traces.

Trace variants: Fig. 4.11 shows the results of trace variants with simulated testing. For example, we consider each trace in reverse (noted by Rev). We also explore starting at different points in the flight (i.e., starting at the 1Q, 2Q, and 3Q point in the flight) by adjusting our starting position and throughput to the corresponding time-step in the trace and then completing a full loop. This works out to the same number of time-steps as the original trace, with location variance relative to the beginning and end of the flight (e.g., starting and ending closer or further from the GCS, or in a different orientation). Chimera performs well in all cases, with the relative performance of schemes following the same trends as before in our initial testing. For the 2Q **FL1** variant, Flight Agnostic performs slightly better than Chimera. Here, the flight begins in an area of poor throughput which gets better, leading Chimera to be conservative in bitrate allocated to live video. Flight Agnostic is also slightly more conservative, but its model ignores the initial data after 5 epochs, whereas Chimera continues to utilize this data in its throughput model.

Synthetic traces: We generated 100 synthetic traces using the methodology described in §4.4, which were based on a flight path that spans 0.5 to 7.5 miles (like the FL datasets). Fig. 4.12 shows a Cumulative Distribution Function (CDF) of the reward across the different traces under test. Chimera out-performs Flight Agnostic in all cases, and performs much

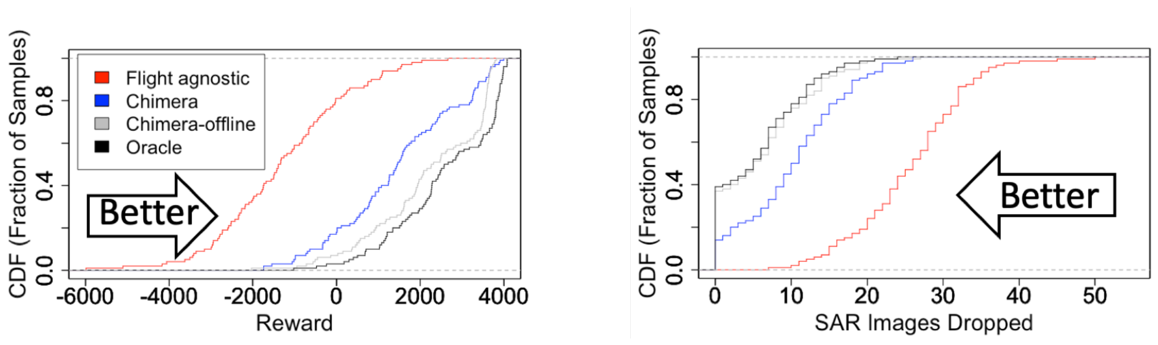


Figure 4.12. Effectiveness of Chimera with 100 synthetic trace tests. The left figure shows the reward comparisons, and accompanying SAR drops are shown on the right side

closer to the Oracle scheme. As expected, Chimera-offline (where the models are pre-trained on data offline, prior to flight) performs slightly better than Chimera because of the advantage of the models being pre-trained on the test data and robust error and throughput models being available immediately, rather than having to take time to be learnt online.

Chimera deep dive: To get a better sense of how Chimera improves performance, Fig. 4.13 presents time series plots to depict how Chimera, Flight Agnostic, and the Oracle scheme make decisions based on the data and state for the **FL1** trace, in a simulated test. Each pair of graphs corresponds to a scheme, with the left graph showing the actual throughput in the trace (black line) and the maximum live video bitrate ($\bar{V}_n^{max}$) allowed by the scheme at each time-step (note that the actual live video bitrate could be lower to account for short-term network fluctuations, calculated as (4.2)). The right graph shows the number of untransmitted SAR images (SAR buffer) for each scheme (the black dotted line indicates where SAR images become stale and are dropped). We see Flight Agnostic is more aggressive with its live video bitrate and does not properly prepare for periods of poor throughput. Whereas, Chimera and the Oracle account for future periods of lower throughput by throttling back the maximum permitted live video bitrate. Consequently, the SAR buffer size (number of untransmitted images) fills up faster for the Flight Agnostic with 23 images being dropped, while only 8 and 1 images are dropped with Chimera and the Oracle, respectively.

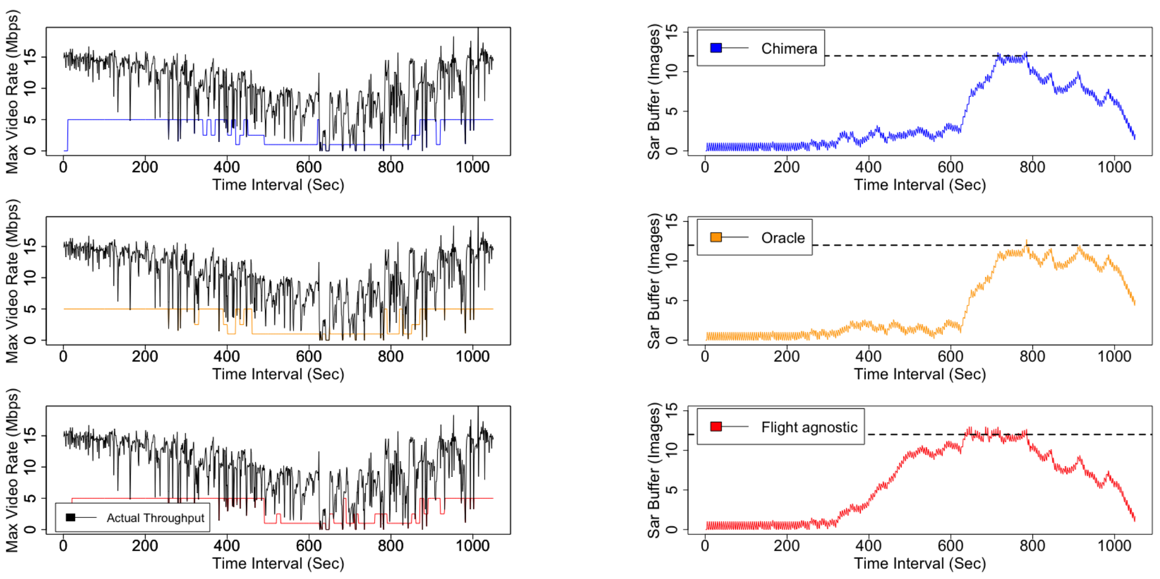


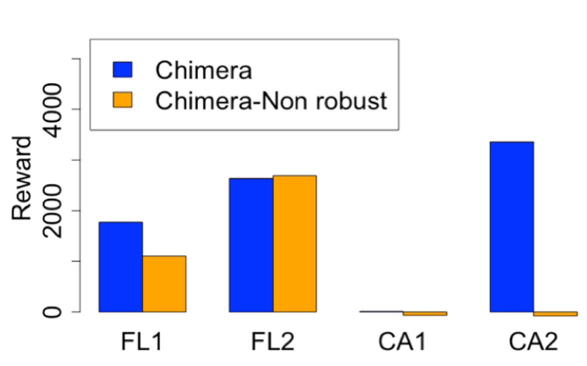
Figure 4.13. Illustrating how Chimera improves performance with video (left) and SAR (right). The top pair of graphs corresponds to Chimera, the middle pair to the oracle, and the bottom pair to Flight Agnostic

4.5.3 Evaluating design variants

We next evaluate variants of Chimera with simulated tests in order to both explore the importance of some of Chimera's decisions and also understand the trade-offs involved with alternative choices.

Importance of considering prediction error: Chimera trains and uses a probabilistic error model to account for prediction errors (4.3.5). Fig. 4.14 (left) compares Chimera and a variant, *Chimera-Non robust*, which does not account for prediction errors. The performance is improved when using an error model, with the benefits being particularly significant in the **FL1** and **CA2** traces. For **FL2** alone, adding the error model results in a slight reduction in performance since the algorithm is a bit more conservative in terms of the maximum live video bitrates permitted. For this trace, throughput is both consistent and also plentiful enough that the prediction errors can be recovered from and are not enough to cause SAR drops.

Online learning variants: We compare Chimera to an orientation agnostic variant of online learning that does not consider UAS orientation in the network and error models. Fig.



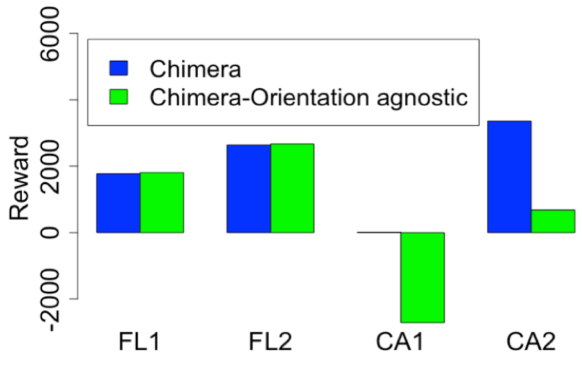


Figure 4.14. Showing the benefits of modeling the prediction errors with Chimera (left), and orientation sensitivity (right)

4.14 (right) shows a comparison of the rewards. We see the FL dataset tests are comparable. CA performance is much higher when considering orientation because (i) the throughput in these traces is more sensitive to orientation and (ii) the traces have more loops, allowing the network models to fully train with each orientation and then utilize the models in the subsequent loops. We also considered variants where the switch to a new orientation model was delayed several epochs. This allowed time for the new orientation model to ingest data prior to use, but we found the tests were inconclusive in providing a clear direction for higher performance.

Facilitating learning with prior data: We explore the feasibility of training a model in one flight and using this model for another flight in a similar environment. To test, we trained a model learned from the actual flight throughput trace, and used this model to test with multiple synthetic traces. We saw benefits to using a prior model – e.g., the average reward improved from 1388.44 to 2377.76, an increase of 71.25%, with **FL1**. We also tested how online learning improves over multiple loops in the flight path, common for many scenarios. To test, we appended a synthetic trace loop to our original trace and compared the performance over two loops with the single loop (with the reward multiplied by 2 for relevant comparison). We repeated this test over 5 iterations with different appended traces and saw an improvement in reward by an average of 9.3% compared to the reward for the regular trace, showing Chimera can improve performance by utilizing models trained from different flights in similar environments.

Look-ahead sensitivity: So far, we have evaluated Chimera with its look-ahead lasting the full flight duration. We explored Chimera's reward with different look-ahead windows. Our results concluded that increasing the look-ahead provides benefits up to 25 epochs (250 seconds), and the benefits diminish beyond this point. This indicates that while a longer look-ahead window is still useful, there is opportunity to reduce Chimera's computational requirements by reducing the look-ahead window, while also enabling Chimera to better adapt to future flight path changes (e.g., emergency situations). We also tested Flight Agnostic and found that even with an unlimited look-ahead, it performed worse than Chimera – e.g., for **FL1**, the reward was lower by 761.46 relative to Chimera, and for the reverse direction of the trace, lower by 2537.45 relative to Chimera.

Impact of epoch size: The choice of epoch size impacts Chimera's performance. Since the maximum permitted live video bitrate does not change during an epoch, a larger epoch size reduces Chimera's ability to adapt. On the other hand, with a larger epoch size, the DP is run less often, and it also takes less time since there are fewer iterations to run. We tested with epoch sizes of 1, 5, 10, and 30 seconds, and found Chimera performs well at 5 and 10 seconds, with performance of 1993.71 and 1772.85, respectively, for the **FL1** trace. Performance degraded at 30 seconds, although Chimera with a 30 second epoch still outperformed the Flight Agnostic scheme significantly. Further, epoch sizes of 1 second are infeasible due to the computational requirements in our test scenarios.

We also measured computation times of the DP on a MacBook Pro. The average DP run time across 10 trials is 6.484, 1.357, 0.700, and 0.210 seconds for epoch sizes of 1, 5, 10, and 30, respectively, indicating an epoch size of 5 or 10 seconds strikes a good balance between Chimera's responsiveness and computation needs. In addition, computational requirements can be further reduced by limiting the look-ahead.

Varying reward function: We explore Chimera's ability to work with different reward functions by changing the parameter W, which captures the importance of SAR relative to video. Recall that we have so far used $W = 8$ which indicates that the penalty of dropping all SAR images is $8\times$ higher than the reward of seeing video at the highest bitrate throughout the flight. We experimented with weights of $W = 2$ and $W = 12$, which decrease and increase the importance of SAR relative to video, respectively. Chimera significantly out-performed Flight Agnostic, amounting to an average reward increase of 2980.65 for Chimera in comparison to Flight Agnostic for **FL1**, showing robustness and improvement with Chimera even with smaller and also more severe SAR drop penalties.

4.6 Related work

UAS Data Transmission: Recent papers [15], [17], [41] explore UAS video streaming for video on demand settings using Adaptive Bitrate (ABR) algorithms. All these works use past network performance (e.g., throughput of the last few video chunks) for future throughput prediction (similar to Flight Agnostic). In contrast, we focus on *live video*, and joint transmission with SAR sensor data. Further, we develop models that predict future throughput based on flight path, complemented with an error model.

Uses of wireless sensor networks for UAS surveillance with path planning is studied in [3], [81]–[83]. These studies focus on discrete connectivity and sensor data sizes rather than the dynamic throughput of real-world UAS networks and variable sensor data. In contrast, we focus on how to predict network throughput, and effectively transmit heterogeneous sensor data, given a long-range UAS flight path.

UAS Communication and Networking: Recent work [29] analyzes hobby UAS flight data and discusses how to generate traffic unique to these settings. However, this work encompasses limited distances based on WiFi, and only works for a few specific and proprietary types of UAS. Wireless UAS networking has been the focus of recent work [30]–[33], [35], [43], which explore the dynamic nature of UAS communication networks, but at shorter distances (typically within VLOS) and with single data types. In contrast, our work is supported by real-world UAS flight data at distances exceeding VLOS, a relatively unexplored area of research due to the difficulty of data collection at such distances. There has been work on modeling UAS communication channels for data transmission [6], [77], [84], reinforcing our theoretical observations (e.g., the effect of distance to throughput). The papers [4], [5], [16], [19] provide high-level information about the challenges and open problems in UAS communication networks, but lack actual flight test data or working solutions to the problems discussed.

Internet video: Much recent work has focused on efficiently delivering Internet video through the design of ABR algorithms [36]–[40], [50], considering video conferencing challenges [78], and exploring the simultaneous transfer of live and time-shifted video [85]. These works do not account for challenges unique to UAS settings, which is our focus.

4.7 Conclusion

In this Chapter, we have designed Chimera to exploit knowledge of the UAS flight path in order to improve transmission decisions for multiple types of heterogeneous sensor data from the UAS to the GCS. We have made the following contributions:

First, we have provided additional characterization and analysis from our two real-world UAS flight datasets (§2), providing deeper insight and motivation into the significant opportunity to optimize data transmission in UAS settings by exploiting knowledge of UAS flight paths, which are typically known in advance of flight.

Second, we have presented Chimera, a system for simultaneously transmitting heterogeneous sensor data with different timeliness and reliability requirements by taking advantage of UAS flight path information. As part of Chimera, we have developed robust models grounded in real-world data that relate UAS network throughput to flight path. Chimera uses an optimal control framework, performing online optimization and augmented with a robust prediction and error model in planning its heterogeneous data transmissions.

Third, we evaluated Chimera using a combination of simulation and emulation experiments, and with multiple real-world flight and synthetic traces generated using a methodology that we developed and validated. Our results show that Chimera offers significant improvement over an approach that does not exploit flight path information. Specifically, in evaluation on our emulation test-bed, Chimera is able to reduce penalties related to dropped SAR image transmissions by $72.4\% - 100\%$ relative to a *Flight Agnostic* scheme, and achieve comparable video qualities of 90.5%, with only minimal increase in SAR images dropped, compared to a perfect Oracle (optimal) scheme that knows future throughput.

5. CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusions

Unmanned Aerial Systems (UAS) are increasingly used to perform sensing and data-gathering in a variety of scenarios (§1). We have taken several key steps towards enabling UAS sensor data transmission applications to efficiently and optimally operate at distances exceeding VLOS, a relatively unexplored, but very important, area of research. UAS operations at long-range distances are of increasing importance due to the opportunities that these extended ranges provide, and regulations are beginning to support long-range flights worldwide [1], [10], [11]. In our work, we have made three contributions.

First, we have flown, collected, and analyzed flight data from real-world UAS flights at distances exceeding VLOS. Our tests range well beyond the existing research in this area (limited to less than 0.25 miles), and provide new insights into how the UAS flight path impacts network throughput [42]. Our flight measurements were taken with a combination of both multirotor and fixed wing UAS, using Tactical Radios, and with omnidirectional and directional antenna configurations. Our results reveal how network throughput depends not only on UAS distance, but also more interestingly on the orientation of the UAS relative to the GCS.

Second, motivated by our measurement results, we designed Proteus, the first system for video streaming at long-range UAS distances, extending the range of UAS video streaming to the edge of connectivity [48]. At this range, our measurements showed extended dropouts make it challenging to simultaneously achieve sub-second video streaming latency and avoid significant loss of video content. However, our data showed the potential to achieve video streaming with modest delays (e.g., tens of seconds). 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. 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.

Third, we have shown through a characterization of two real-world UAS flight datasets that there is significant opportunity to optimize data transmission in UAS settings by exploiting knowledge of long-term UAS flight paths. Based on this observation, we have presented Chimera, a system we designed to simultaneously transmit heterogeneous sensor data with different timeliness and reliability requirements by taking advantage of UAS flight path information. As part of Chimera, we have developed models grounded in real-world data that relate UAS network throughput to flight path. Chimera uses an optimal control framework, performing online optimization augmented with a robust prediction and error model for planning its heterogeneous data transmissions. We evaluated Chimera using a combination of simulation and emulation experiments, and with multiple real-world flight traces and synthetic traces generated using a methodology that we have developed and validated. Our results show that Chimera offers significant improvement over an approach that does not exploit flight path information. Specifically, in evaluation on our emulation test-bed, Chimera is able to reduce penalties related to dropped SAR image transmissions by $72.4\% - 100\%$ relative to a *flight agnostic* scheme. Further, Chimera is able to achieve comparable video qualities of 90.5%, with only a minimal increase in SAR images dropped, compared to a perfect Oracle (optimal) scheme that knows future throughput.

5.2 Future directions

This thesis takes the first step in building robust UAS sensor data transmission applications that account for UAS flight path and can operate at distances exceeding VLOS. Given the importance and growth of UAS applications, and high probability of expanded operations in the near future, there are several avenues that warrant consideration for potential future directions to build upon this research.

Expansion to multiple UAS: Our work focuses on initial characterization and applications of missions that operate with a UAS-to-GCS configuration. While this is a very common configuration, multiple UAS (e.g., swarms) further increase the capabilities and opportunities to enhance mission effectiveness (e.g., they can utilize more sensors and cover more area in less time) [86]. However, multiple UAS networking changes the point-to-point network configuration that our work includes into a large network of interconnected nodes. The expansion of our research into multiple UAS networks is not trivial, but would represent a leap forward and is a next logical step that could build off our measurement insights in exploiting the UAS flight path, and control theoretic approaches to optimize data transmission.

Explore interoperability with satellite communication (SATCOM): Our research focuses on point-to-point communication with the radios having a direct path of communication, where SATCOM is not used or needed. Many UAS operations benefit from having a communication link that can expand to space to reach anywhere in the world [87]. In this manner, a hybrid network of Tactical Radios and SATCOM communication points could be connected to provide UAS sensing capabilities with worldwide reach. This could further expand to provide additional sensor data from UAS across the world and fuse data in a controlled manner.

Experiment with Proteus and Chimera in 5G settings: 5G wireless is an area of high interest and growth, providing millimeter waveforms and increasing throughput. However, the pros of 5G are not without challenges, such as weather, range, and obstruction considerations. There is much ongoing research for 5G, including wireless usage with UAS [88]–[90]. The control frameworks and algorithms we have designed (Proteus and Chimera) would benefit from experimentation with 5G settings for UAS flight - potentially increasing the viability of higher video bitrates (for Proteus) and even more types of sensors (Chimera).

Explore enhancements using artificial intelligence: Our research built throughput prediction and error models through detailed analysis and insights gleaned from real-world flight datasets. Additionally, we explored the implications of learning these models online, and using them in UAS sensor data transmission algorithms. Since real-world flight data is challenging to collect, especially at extended distances, it may be beneficial to explore using artificial intelligence to enhance the models. One specific area of potential pursuit would be to explore model-based reinforcement learning in the context of UAS data transmission systems.

REFERENCES

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

[2] 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.

[3] Y. Chen, H. Zhang, and M. Xu, "The coverage problem in UAV network: A survey," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, IEEE, 2014, pp. 1–5.

[4] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[5] 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.

[6] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on UAV communications for 5G and beyond," *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, 2019.

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

[8] R. Dunkel, R. Saddler, and A. Doerry, "Use of unmanned SAR and EO/IR sensor suites for monitoring wildfires," in *Radar Sensor Technology XXI*, International Society for Optics and Photonics, vol. 10188, 2017, 101881F.

[9] Y. Ban, P. Zhang, A. Nascetti, A. R. Bevington, and M. A. Wulder, "Near real-time wildfire progression monitoring with sentinel-1 SAR time series and deep learning," *Scientific Reports*, vol. 10, no. 1, pp. 1–15, 2020.

[10] 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/.

[11] *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.

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

[13] 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.

[14] 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.

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

[16] O. K. Sahingoz, “Mobile networking with UAVs: Opportunities and challenges,” in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2013, pp. 933–941.

[17] S. Kacianka and H. Hellwagner, “Adaptive video streaming for UAV networks,” in *Proceedings of the 7th ACM International Workshop on Mobile Video*, 2015, pp. 25–30.

[18] F. L. Templin, R. Jain, G. Sheffield, P. Taboso-Ballesteros, and D. Ponchak, “Requirements for an integrated UAS CNS architecture,” in *2017 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, IEEE, 2017, 2E4–1.

[19] H. Nawaz, H. M. Ali, and A. A. Laghari, “UAV communication networks issues: A review,” *Archives of Computational Methods in Engineering*, pp. 1–21, 2020.

[20] *14 CFR: Aeronautics and Space*, Code of Federal Regulations, 2019.

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

[22] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, “Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges,” *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[23] P. Hügler, F. Roos, M. Schartel, M. Geiger, and C. Waldschmidt, “Radar taking off: New capabilities for UAVs,” *IEEE Microwave Magazine*, vol. 19, no. 7, pp. 43–53, 2018.

[24] IMSAR, *Synthetic aperture radar*, 2020. [Online]. Available: https://www.imsar.com/portfolio-posts/synthetic-aperture-radar/.

[25] S. Manfreda, M. F. McCabe, P. E. Miller, R. Lucas, V. Pajuelo Madrigal, G. Mallinis, E. Ben Dor, D. Helman, L. Estes, G. Ciraolo, *et al.*, “On the use of unmanned aerial systems for environmental monitoring,” *Remote sensing*, vol. 10, no. 4, p. 641, 2018.

[26] G. Stimson, *Introduction to Airborne Radar*. SciTech Publishing Inc, 1998.

[27] A. W. Doerry and F. M. Dickey, “Synthetic aperture radar,” *Optics and photonics news*, vol. 15, no. 11, pp. 28–33, 2004.

[28] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, “A tutorial on synthetic aperture radar,” *IEEE Geoscience and remote sensing magazine*, vol. 1, no. 1, pp. 6–43, 2013.

[29] A. Baltaci, M. Klügel, F. Geyer, S. Duhovnikov, V. Bajpai, J. Ott, and D. Schupke, “Experimental UAV data traffic modeling and network performance analysis,” in *Proceedings of the 40th IEEE International Conference on Computer Communications*, 2021.

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

[31] 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.

[32] G. Geraci, A. Garcia-Rodriguez, L. G. Giordano, D. López-Pérez, and E. Björnson, “Understanding UAV cellular communications: From existing networks to massive MIMO,” *IEEE Access*, vol. 6, pp. 67 853–67 865, 2018.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] 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.

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

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] M. Nekrasov, R. Allen, and E. Belding, "Performance analysis of aerial data collection from outdoor IoT sensor networks using 2.4 ghz 802.15. 4," in *DroNet 2019*, 2019, pp. 33–38.

[47] 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.

[48] R. Shirey, S. Rao, and S. Sundaram, "Optimizing quality of experience for long-range UAS video streaming," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQoS)*, IEEE, 2021.

[49] 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.

[50] 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.

[51] 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.

[52] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," 6, vol. 36, Elsevier, 2000, pp. 789–814.

[53] I. Burrington, "What happens to the internet after a disaster," 2017. [Online]. Available: http://nymag.com/intelligencer/2017/10/what-happens-to-the-internet-after-a-disaster.html.

[54] M. Zaller, "Fire View Infrared Downlink (FirstNet). Wildland fire common operating picture and interactive drawing," 2017.

[55] "MANET operational field test with the New York City police department emergency service unit," Department of Homeland Security, Tech. Rep., 2017.

[56] "FHWA white paper on mobile ad hoc networks," Department of Transportation, Tech. Rep., 2018.

[57] *Super bowl broadcasters leverage MPU5 for live shots*, 2017. [Online]. Available: https://www.persistentsystems.com/super-bowl-broadcasters-leverage-mpu5-for-live-shots/.

[58] *MPU specification sheet*, 2016. [Online]. Available: http://www.persistentsystems.com/pdf/MPU4%5C_SpecSheet.pdf.

[59] Trellisware, *TW-900/950 TSM shadow radio*, 2020. [Online]. Available: https://www.trellisware.com/wp-content/uploads/2020/09/TW-900-950-TSM-Shadow-Product-Datasheet.pdf.

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

[61] J. P. Rula, J. Newman, F. E. Bustamante, A. M. Kakhki, and D. Choffnes, "Mile high WiFi: A first look at in-flight internet connectivity," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2018, pp. 1449–1458.

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

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

[64] *Bat-4*, 2019. [Online]. Available: http://martinuav.com/portfolio/the-bat-4/.

[65] *Haigh-farr blade antennas*, 2020. [Online]. Available: https://www.haigh-farr.com/docs/default-source/products-data-sheets/general%5C_blade%5C_booklet%5C_rev1.pdf.

[66] 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.

[67] 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.

[68] 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.

[69] 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 *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 183–196.

[70] 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.

[71] 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.

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

[73] 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.

[74] R. Gangula, O. Esrafilian, D. Gesbert, C. Roux, F. Kaltenberger, and R. Knopp, "Flying rebots: First results on an autonomous UAV-based LTE relay using open air interface," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, IEEE, 2018, pp. 1–5.

[75] H. T. Friis, "A note on a simple transmission formula," *proc. IRE*, vol. 34, no. 5, pp. 254–256, 1946.

[76] J. L. Volakis, *Antenna engineering handbook*. McGraw-Hill Education, 2007.

[77] F. Ono, H. Ochiai, and R. Miura, "A wireless relay network based on unmanned aircraft system with rate optimization," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7699–7708, 2016.

[78] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 267–282.

[79] R. B. D'Agostino and M. A. Stevens, *Goodness-of-fit-techniques*. Routledge, 2017.

[80] Google, *Recommended upload encoding settings*, 2021. [Online]. Available: https://support.google.com/youtube/answer/1722171.

[81] P. Sun and A. Boukerche, "Performance modeling and analysis of a UAV path planning and target detection in a UAV-based wireless sensor network," *Computer Networks*, vol. 146, pp. 217–231, 2018.

[82] A. Lamine, F. Mguis, H. Snoussi, and K. Ghedira, "Coverage optimization using multiple unmanned aerial vehicles with connectivity constraint," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2019, pp. 1361–1366.

[83] L. C. B. da Silva, R. M. Bernardo, H. A. de Oliveira, and P. F. F. Rosa, "Unmanned aircraft system coordination for persistent surveillance with different priorities," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, IEEE, 2017, pp. 1153–1158.

[84] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328–331, 2017.

[85] D. Ray, J. Kosaian, K. Rashmi, and S. Seshan, "Vantage: Optimizing video upload for time-shifted viewing of social live streams," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 380–393.

[86] B. Vergouw, H. Nagel, G. Bondt, and B. Custers, "Drone technology: Types, payloads, applications, frequency spectrum issues and future developments," in *The future of drone use*, Springer, 2016, pp. 21–45.

[87] Z. Kaleem and M. H. Rehmani, "Amateur drone monitoring: State-of-the-art architectures, key enabling technologies, and future research directions," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 150–159, 2018.

[88] R. Muzaffar, C. Raffelsberger, A. Fakhreddine, J. L. Luque, D. Emini, and C. Bettstetter, "First experiments with a 5G-connected drone," in *Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, 2020, pp. 1–5.

[89] G. Makropoulos, H. Koumaras, F. Setaki, K. Filis, T. Lutz, P. Montowtt, L. Tomaszewski, P. Dybiec, and T. Järvet, "5G and unmanned aerial vehicles (UAVs) use cases: Analysis of the ecosystem, architecture, and applications," in *Handbook of Research on 5G Networks and Advancements in Computing, Electronics, and Electrical Engineering*, IGI Global, 2021, pp. 36–69.

[90] S. Sekander, H. Tabassum, and E. Hossain, "Multi-tier drone architecture for 5G/B5G cellular networks: Challenges, trends, and prospects," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 96–103, 2018.

VITA

Russell Shirey is a Ph.D. Candidate in the School of Electrical and Computer Engineering at Purdue University, advised by Professor Sanjay Rao and Professor Shreyas Sundaram. His research interests are in Computer Networking Systems and Control Theory. His research focuses on characterization and improvements to networking with Unmanned Aerial Systems (UAS). Specifically, Russell's research uses a combination of control-theoretic approaches and online optimization, with insights gleaned from real-world UAS flight data, in order to improve sensor data transmission from the UAS to the ground. Prior to his doctoral studies, Russell received his M.S. degree in Computer Engineering from the Air Force Institute of Technology, Wright-Patterson AFB, Ohio, and his B.S. degree in Computer Engineering from the University of Michigan-Dearborn, Dearborn, Michigan.