A LOCATION-DEPENDENT RUNS-AND-GAPS MODEL FOR PREDICTING TCP PERFORMANCE OVER A UAV WIRELESS CHANNEL

H.T. Kung*, Chit-Kwan Lin*, Tsung-Han Lin*, Stephen J. Tarsa*, Dario Vlah*,
Daniel Hague†, Michael Muccio†, Brendon Poland†, Bruce Suter†
**Harvard University, Cambridge, MA*
†Air Force Research Laboratory, Rome, NY

Abstract

In this paper, we use a finite-state model to predict the performance of the Transmission Control Protocol (TCP) over a varying wireless channel between an unmanned aerial vehicle (UAV) and ground nodes. As a UAV traverses its flight path, the wireless channel may experience periods of significant packet loss, successful packet delivery, and intermittent reception. By capturing packet run-length and gap-length statistics at various locations on the flight path, this location-dependent model can predict TCP throughput in spite of dynamically changing channel characteristics. We train the model by using packet traces from flight tests in the field and validate it by comparing TCP throughput distributions for model-generated traces against those for actual traces randomly sampled from field data. Our modeling methodology is general and can be applied to any UAV flight path.

1. Introduction

TCP is the most widely-used reliable transport protocol and is employed in many different applications, ranging from Web surfing, to file transfer, to distributed computing [1]. In this paper, we consider using TCP to transport data from ground nodes to a low-flying, fixed-wing UAV over a wireless channel. Naturally, research and application development in this scenario benefits from repeatable experimentation that captures operation in realistic environments. This is best achieved in physical deployments, though repeatability is often limited by economic and logistical constraints. As a result, we are interested in constructing a model for predicting TCP throughput over the wireless channel that will enable repeatable realistic emulation to support application and system development. This model must address the large variations in the wireless channel that the UAV may experience in the course of its flight path, and allow the emulator to use the real TCP stack for congestion control [2].

One of the challenges in capturing channel conditions is to account for physical and environmental effects that may cause packet loss/reception events to be correlated over time [3]; in modeling, this effect is often referred to as *channel memory*. In periods where the wireless channel sees significant packet loss, TCP throughout it largely determined by patterns of consecutive loss (gaps) and consecutive reception (runs) that are driven by channel memory, rather than just the average packet loss rate over a period.

To illustrate why the run and gap pattern is important, we consider TCP behavior in a simple emulation experiment where we fix the average packet loss rate at 10%, while varying the gap and run lengths in the channel. Specifically, let us construct a sequence of packet receptions and losses using alternating segments of $9i$ receptions and i losses, while varying the value of i. Figure 1 demonstrates that different run and gap sizes, arising from varying degrees of channel memory, have a significant impact on TCP performance. Thus, to properly model wireless channels for the purpose of TCP performance evaluation, we ought to take run and gap behavior into account, in addition to first-order statistics such as average packet loss rate.

Looking ahead at an example, Figure 4 (lower portion) in Section 4 depicts the measured packet reception of a UAV receiving transmissions from ground nodes over the course of its flight path. In keeping with our interest in employing commercial-off-the-shelf (COTS) components [4], we use the IEEE 802.11 medium access control and physical layer [5] throughout this work. The large variations in loss behavior that we observe with respect to the position of the UAV suggests a location-dependent model is necessary, where distinct submodels are used to characterize different legs of the flight path. Practically, we will have to train each submodel with a relatively small amount of data, as we are subject to the velocity of and area covered by the plane.

Our modeling approach can meet these objectives. In the past, other modeling efforts employed multi-state Markov chains, or simply drew run and gap lengths from a distribution fitted to measured data (we review these methods in Section 2). However, we will show that these existing methods can be further improved by grouping contiguous locations with similar run and gap statistics (referring to the combined portions of the flight path as legs), and then clustering within legs based on run/gap distributions. We train a different submodel for clusters within each leg, which can later be combined to produce accurate trace-based TCP throughput predictions in the presence of limited training data.

2. Related Work

Trace-driven emulation (see, e.g., [6], [7]) of the wireless channel typically entails recording representative traces from a physical deployment, training a model based on the traces, and then using the model to generate synthetic traces to drive emulation. Commonly, these synthetic traces are replayed to exercise a higher-layer protocol under test (e.g., TCP, in this paper).

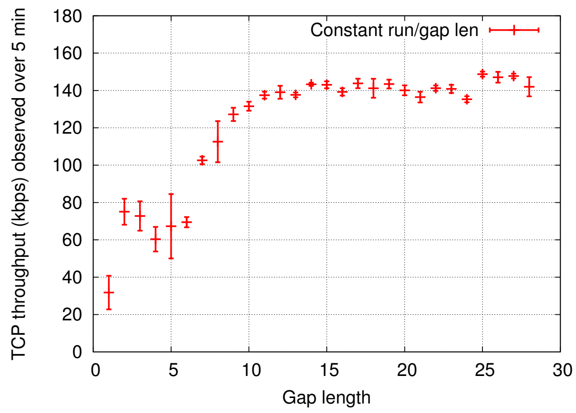


Figure 1: TCP throughput depends on run/gap sizes, rather than the mean packet loss rate. We measured throughput of a TCP connection over an emulated channel with a constant 10% mean packet loss, but increased run/gap sizes. Maximum achievable throughput without losses is 160 kbps. Each data point is an average of 5 runs, each lasting 5 minutes; error bars show the standard deviation. Notice an increase in TCP throughput as run/gap sizes increase, while the mean packet rate is kept the same.

Many link-layer trace models decompose into submodels capturing different facets of channel behavior, such as the classic two state Gilbert [8] and Fritchman models [9]. Nguyen et al. [10] also use two submodels, capturing error and error-free packet bursts based on burst length distributions. Their method is to fit piecewise curves to model the two distributions independently. This approach shows improved accuracy in predicting TCP throughput over prior loss models.

Konrad et al. present a Markov-based Trace Analysis (MTA) algorithm [11] that treats packet traces collected by mobile GSM nodes as being generated by a process with two states: lossy and error-free. MTA partitions the collected traces into these two states, and again estimates length distributions for subtraces in each state. For the lossy state, they then train a discrete time Markov chain (DTMC), capable of generating synthetic lossy subtraces. All three components of the model are combined to output full synthetic packet traces by alternately drawing lengths from lossy and error-free length distributions, and emitting symbols according to the DTMC for the lossy state. MTA-generated traces show improved error burst statistics over simpler models, however the algorithm does not take into account the mobile node's location or trajectory, and relies on DTMCs of order 6 to produce accurate results, which require large training data sets.

Previous work [12], [13] on the Lakehurst MANET trace dataset has implied the importance of incorporating location dependence into channel modeling approaches. However, the volume of data needed can make this impractical for trace-based methods. As an alternative, radio propagation models are sometimes used [14], [15], [16], though realism comes at the expense of arduous model tuning. Instead, our location-dependent methodology, with its use of trace clustering, focuses on producing accurate results on relatively little training data.

3. Model Overview

As shown in Figure 1 and argued recently by Gurtov and Floyd [15], the second-order statistics of the wireless channel (run and gap length distributions) severely impact emulated/simulated TCP behavior. Therefore, we focus on accurate representation of these properties over the entire UAV flight path by first breaking up a mobile trace into more granular *locations*. Because this approach results in having a larger number of location-dependent models to train and less training data per-model, our method must be able to train accurately on relatively limited trace data.

To accomplish this, we cluster fixed-sized blocks of traces by both position and loss behavior. Each block is registered to one of 45 locations along the flight path based on UAV telemetry data. The choice of block size and number of locations is discussed in Sections 4 and 5. We then combine contiguous locations whose blocks exhibit similar mean gap lengths into *legs* of the flight. Within each leg, we cluster together individual blocks based on their mean gap-lengths, noting the proportion of traces falling into each cluster. An order-n Markov model is trained per cluster in each leg.

Grouping locations into legs based on gap behavior improves model accuracy by making more training data available to model each location within a leg. Further clustering within each leg results in lower error by reducing the variance that must be explained by any single model. Figure 2 summarizes this structure.

Each cluster's Markov model outputs a synthetic, block-length trace. For a particular leg, we create a full-length synthetic trace by concatenating block-length traces. submodels. The sequence of the cluster submodels is randomly chosen based on the observed cluster distribution. Figure 3 summarizes this process for a given flight path.

Our approach departs from prior work in that it treats trace data from different physical locations separately when their channel characteristics are very different, but pools them together when characteristics are similar. This means that a model for one portion of the flight path need not explain starkly different behavior observed in another portion of the flight path. Mean-gap clustering of individual blocks allows us to have accurate low-order Markov models, such as models of order 2, that are trainable with a relatively smaller set of trace data. This is in contrast to the MTA algorithm, where models of order 6 are used and trained on a large dataset. In that approach, high-complexity modeling may be necessary because a single model is trained on all traces from a mobile node, regardless of position or loss behavior.

4. Trace Collection in the Field

We collect packet traces of the ground-to-air wireless channel by recording individual packet transmissions and receptions between stationary ground transmitters and mobile receivers aboard an overhead UAV. A similar approach has been applied in the

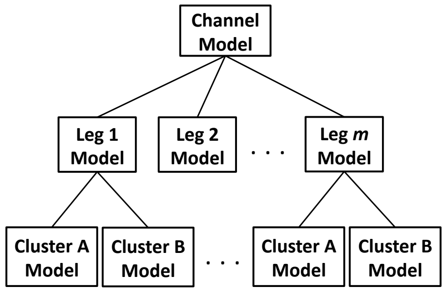


Figure 2: The decomposition of our wireless channel model into component submodels.

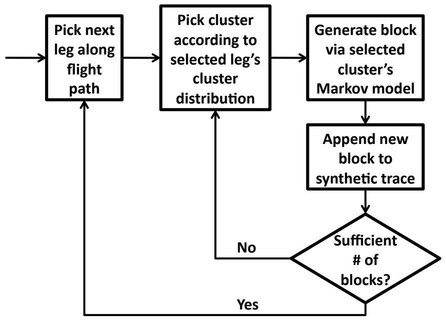

Pick next leg along flight path
Pick cluster according to selected leg's cluster distribution
Generate block via selected cluster's Markov model
Append new block to synthetic trace
Sufficient # of blocks?
No
Yes

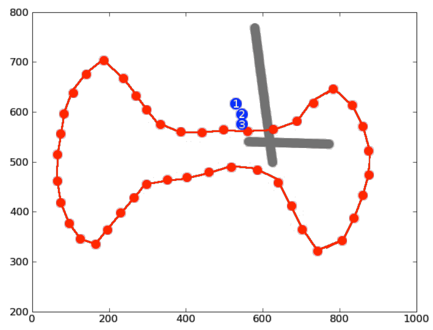
Figure 3: The process our model employs to generate a synthetic packet trace along the modeled flight path.

air-to-ground direction in ongoing research. In this section, we describe the equipment, facilities, experimental setup, flight plan, and post-processing required to construct these traces.

4.1. Equipment and Facilities

Both transmitting and receiving nodes are Mobile Internet Devices (MIDs) based on 800MHz x86-compatible Intel Atom processors, with Marvell SD8686 802.11b/g internal SDIO radios. Three transmitting MIDs are placed on the ground spaced approximately 20m apart, and four receiving nodes are mounted on the UAV (see Section 4.2). During flight, ground transmitters broadcast 1420-byte UDP packets, while airborne receivers log the sequence number and sender ID of each arriving packet. Results presented in this paper are based on a transmitter rate of 10 packets per second, however the same methodology has been applied at rates of up to 200 packets per second. We rely on CSMA to provide fair share of the wireless channel among transmitters, although we note that a round-robin scheme is an alternative.

The UAV is a SIG Rascal 110 with a 110" wingspan. The fuselage is comprised of a balsa wood airframe covered by a fiberglass cowl, with fixed landing gear and wing struts made of aluminum. The plane's engine



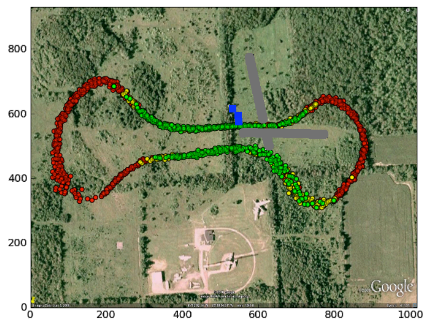


Figure 4: Upper: Cyclic flight path taken by the UAV. Each flight consists of approximately 10 clockwise laps. Ground transmitter locations are in blue; runway positions are in grey. Units are in meters. Lower: Overlay of packet reception data between ground TX1 and airborne, wingtip-mounted RX1, aggregated over all flights. Each dot represents a 20-packet block of the trace. Blocks with a mean gap length ≤ 5 are in green, blocks with mean gap between 5 and 19 are in yellow, and blocks with no packet reception are in red.

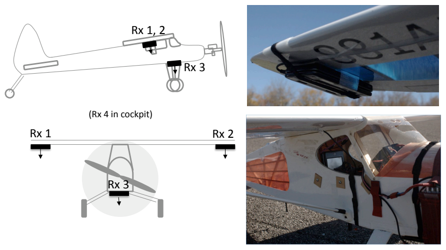


Figure 5: The positions of externally mounted MIDs on the UAV. The inset photos show a closeup of the right wing-mounted MID, as well as the cockpit-mounted MID.

is electric, and is powered by batteries held in saddle bags draped over the exterior of the nose of the plane. The electric engine simplifies aircraft maintenance due in large part to its cleanliness as compared to gas engines. Though the engine need not be refueled, batteries must be swapped every 30 minutes, limiting flying time conservatively to 20 minutes. Servicing time between flights requires a minimum of 5 minutes to replace batteries, download data, and set up new experiments. UAV battery packs are also used to power onboard MIDs, eliminating the need to monitor and charge the MIDs' individual batteries.

The airfield is located atop a large hill in Stockbridge, NY. Facilities include two runways, a hangar, an electronics workbench with a reserve-battery charging station, a bunker capable of monitoring surrounding airspace and weather, and a trailer for technicians and pilots. The property is free of 802.11 radio signals not belonging to flight experiments. The surrounding valleys are primarily woodland and farmland. Flights were conducted in late October and early November, after the leaves of deciduous trees had fallen.

For UAV flight, three personnel are required: a pilot, a spotter, and a technician to monitor instruments and autopilot status from the ground. Takeoff, taxi, and landing are conducted by a pilot via remote joystick, and control of the plane is handed over to an autopilot system when the aircraft reaches cruising altitude. While the plane is being serviced between flights, a control laptop is used to download data from each MID and to issue commands for subsequent flights.

Via autopilot, the UAV is capable of following a flight path passing through preset waypoints. This allows us to collect data over multiple laps within a flight, and to repeat multiple flights as necessary. In our tests, on a day with low wind, the autopilot achieves repeatably high positional accuracy across laps. Over all flights, positional deviation with respect to a fixed way point was no more than 13m when the plane was level, and no more than 25m when the plane was banking through a turn. Assuming that the aircraft can correct its course within 5 seconds of the initial departure from the flight path, at an airspeed of 20m/s, and with a link rate of 10 packets per second, this deviation results in a difference of 2 and 6 packet transmissions measured during that portion of the flight cycle, respectively.

Throughout the flight, the autopilot system records instrument readings approximately every 250ms. For our purposes, the most relevant readings are altitude, airspeed, GPS latitude, GPS longitude, and GPS clock-time.

4.2. Experimental Setup and Flight Path

MID transmitters were placed on the ground in a field with their screens facing skyward, as depicted by the blue circles in Figure 4. Three receivers are strapped to the exterior of the plane: one on the underside of each wingtip, and one between the landing gear struts on the underside of the fuselage, as shown in Figure 5. These three MIDs are oriented with their screens facing downward toward the ground. An additional receiver is placed in the cockpit of the plane with its screen pointing forward toward the propeller, as depicted in the inset photo in Figure 5.

The flight path is a dumbell-shaped cycle (Figure 4) that passes beyond the extremes of MID radio range. Each lap of the dumbell lasts approximately 90 seconds, with the UAV autopilot set to maintain an air speed of 20m/s and an altitude of 75m. Cruising time for each flight is about 15 minutes, meaning the UAV executes 10 laps. Our complete set of traces spans six flights collected over five hours in one day.

4.3. MID Antenna Patterning

In order to characterize the ground-to-air wireless channel in terms of transmitter/receiver positions, we also must account for the effects of MID orientation on packet reception. To do this, we mapped the MID's antenna pattern in an anechoic chamber. An MID was attached to a fiberglass positioning arm and Received Signal Strength (RSS) was measured as the arm was rotated about its three primary axes.

In Figure 6, we show the signal strength for rotations about two axes. The upper plot shows the RSS pattern that the cockpit-mounted receiver would experience with respect to ground transmitters throughout the flight. In the lower plot, angles between 0° and 180° describe the RSS pattern for exterior-mounted MIDs with respect to ground nodes. Note that though the MIDs on the exterior of the plane will enjoy little degredation in signal strength, we expect the MID in the cockpit will see relative fluctuations of up 15dB, depending on whether a ground node faces the front or back of the receiver. As we discuss in Section 4.4, this effect is exhibited in Figure 7.

4.4. Post-processing: Positional Traces, Locations, and Legs

After synchronizing MID clock readings to GPS time, we use packet timestamps to establish the plane's latitude and longitude at each reception event from telemetry data. In Figure 7, each blue dot shows the position of the UAV when a packet was received, for each transmitter-receiver pair (the flight path is clockwise). We note that the reception of the cockpit MID (RX4) degrades as soon as it passes the ground nodes, in both directions, as predicted by its antenna pattern and orientation (see Figure 6). In contrast, the

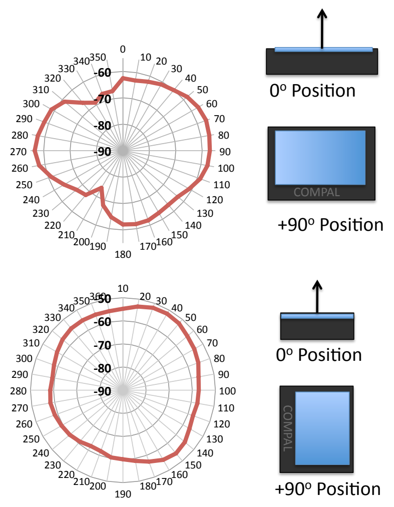


Figure 6: MID antenna patterns collected in an anechoic chamber. Radial plot of RSS in dB, as the receiver is rotated about a single axis. Upper: RSS pattern of the cockpit-mounted receiver with respect to fixed ground transmitters; angles between 180° and 360° apply to packets received from a transmitter as the UAV passes over and departs from the node. Lower: RSS of externally-mounted MIDs with respect to fixed ground nodes is restricted to angles between 0° and 180°.

orientations of RX1-RX3 allow them to receive on approach as well as departure.

In order to build a contiguous packet trace of each flight, we must also associate packet *losses* with UAV position. We calculate the number of lost packets between receptions by subtracting consecutively received sequence numbers and linearly interpolate timestamps between the two received packets. This allows us to look up the plane's position using the interpolated timestamps, and to construct a contiguous trace where each packet transmission event is mapped to positional coordinates for the UAV, regardless of actual reception.

Next, we associate blocks of a trace with discrete *locations* along the flight path. By discretizing the path and registering blocks to the closest location, we ensure that there is sufficient data to support modeling at positions covering legs of interest. We define 45 locations evenly spaced along the flight path, meaning that on average, a block of 20 packet events will be collected for each location during every lap. Traces are divided into blocks of length 20, and each block is labeled with the GPS coordinates of its 10th packet. Deviations in the flight path will cause blocks in consecutive laps to be misaligned, so we use a registration window around each location to associate each individual block with its nearest location by position.

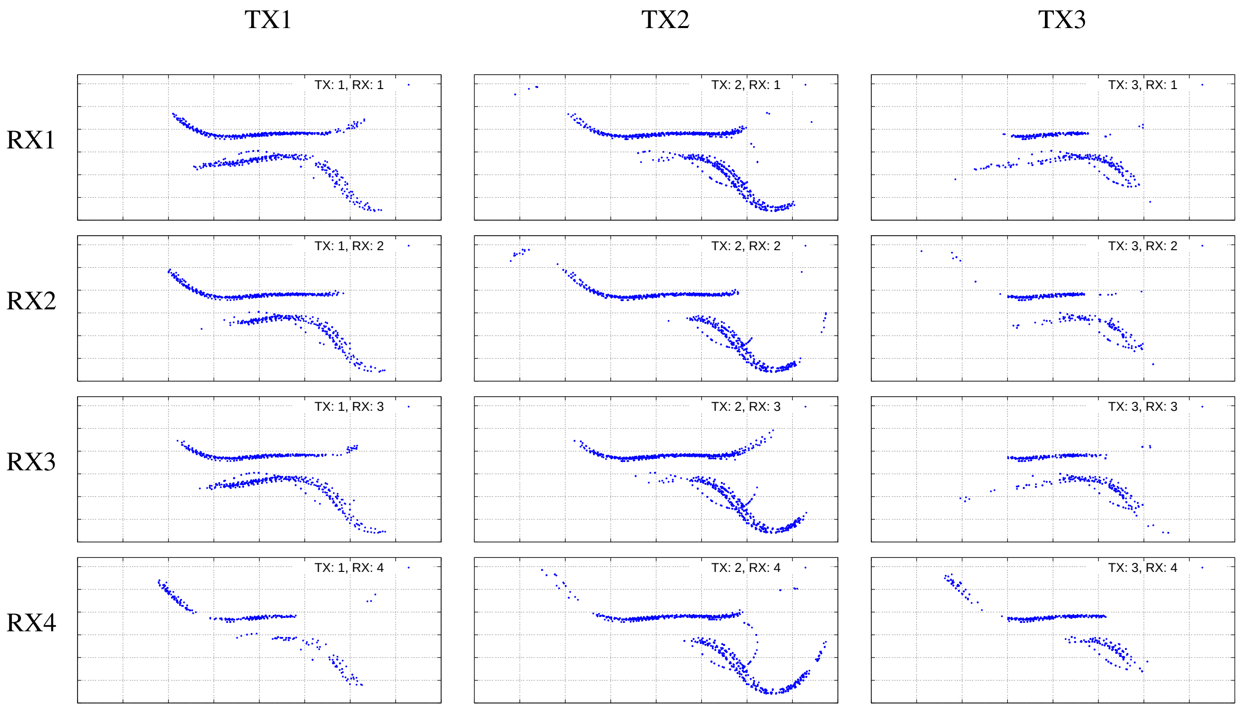


Figure 7: Packet receptions observed for different transmitter-receiver pairs, plotted at the location of the UAV at the time of reception. RX1-RX3 are mounted on the UAV exterior and RX4 is mounted in the cockpit.

5. Model Training Methodology

5.1. Trace Clustering

A block is the basic unit for clustering on channel quality, and hence it has to be sufficiently long to obtain channel quality statistics, but not be so long as to preclude clustering at a useful granularity. The selected block size of 20 packets appears to be a reasonable choice given the expected variations in channel quality and deviations in autopilot-controlled flight trajectory. Choosing the length of a leg is also an engineering decision; short legs allow the model to give information about more locations with finer granularity, but the reduction in training data per leg may result in less accurate models.

Within each leg, we cluster blocks by their mean gap length, a metric that reflects channel quality. We choose gap length instead of run length because a run can easily be interrupted by a single packet loss even if the channel quality is relatively good, making run lengths a less stable metric. The k-means algorithm [17] is applied to cluster the blocks. A representative clustering is illustrated in Figure 8, where we show the histogram of mean gap length from a typical leg where the UAV is within radio range of the ground transmitter. Two clusters of channel quality can be observed: most of the time, the UAV has a good connection with the ground node, but, occasionally, the channel degrades enough to cause all packets in a block to be lost.

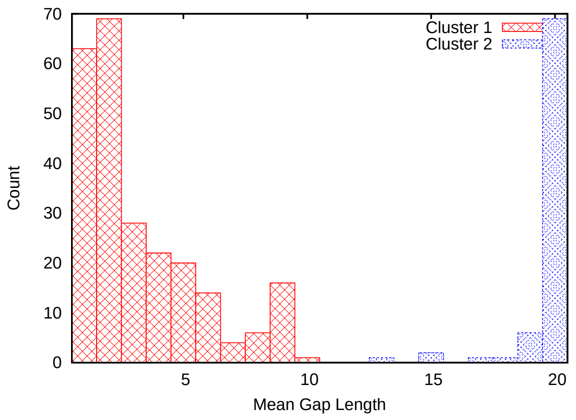


Figure 8: Histogram of mean gap length and corresponding cluster identity of each bin. As in this example, two clusters are usually observed representing good (Cluster 1) and bad (Cluster 2) channel quality. Clustering the blocks helps reduce the model complexity.

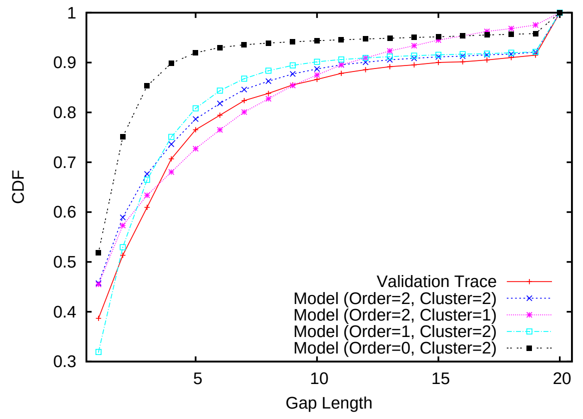


Figure 9: Gap-length CDFs of model-generated synthetic traces compared against those of the validation traces. Increasing both the number of clusters and the order of the model results in a gap-length CDF closer to that of the validation set, provided there is sufficient data to properly train the model.

5.2. Markov Model Training

For each cluster, we train an order-n Markov chain, and combine per-cluster Markov models into a per-leg model by assigning a probability proportional to the cluster size to each (i.e., via the observed cluster distribution).

In our formulation, each state of the Markov model corresponds to a sequence of n symbols in a block, and we train the models by counting the number of transitions between sequences of n symbols occurring in the trace. Note that the size of the training set limits the order of the Markov chain. An order-n Markov chain has 2^{2n} state transitions, so with a leg size of four blocks containing approximately 2400 symbols over 30 laps, it would be unlikely that the $2400/n = 600$ countable transitions are enough to accurately train any model of order $n \geq 4$.

6. Evaluation

6.1. Performance Measurement

In order to evaluate the performance of our models, we split the traces from six flights in two halves. The first half contains traces from the first three flights and is used as a training set, while the other half is used to validate the trained model. We chose to use a single transmitter-receiver pair (TX2 and RX4 as shown in Figure 7) to evaluate the model because it has the most packet receptions. In Section 6.2, the models are evaluated by comparing the run and gap distributions of model-generated traces against those of the validation traces. Additionally, in Section 6.3, we compare TCP throughput for model-generated synthetic traces with that of the validation traces.

6.2. Model Training Accuracy

Figure 9 shows gap length distributions from traces generated by models of orders $n = 0, 1, 2$, using $k = 1, 2$ clusters alongside the corresponding validation-set distribution. For both Sections 6.2 and 6.3, we train a single leg containing 12 consecutive locations, representing the northerly straight leg of the flight path (see Figure 4). Results from the southerly leg have similar performance.

We first note that when using a single cluster, blocks with large gap length bias the model's fit: short gaps with lengths between 5 and 10 are under-represented and larger gaps between 12 and 15 are over-represented, while the model fails to capture blocks with gap length 20. Clustering seperates small gap length blocks from large gap length blocks, and results in a better fit across all lengths. Second, we note that increasing the order of the Markov chain also improves performance, which indicates the existence of channel memory.

Examining error as each model parameter is varied in isolation confirms that order $n = 2$ with $k = 2$ clusters is best for this leg. Tables 1, 2, and 3 compare traces generated by models with different parameters by showing the normalized squared distance between the run- or gap-length CDF of model-produced traces and the run- or gap-length CDF of the underlying field-collected trace. We use this metric as an indication of the relative error produced by different models. The values presented are an average across a large number of runs, and the comparisons hold with a high degree of stability. We see that increasing either the order of the model beyond $n = 2$, or the number of clusters beyond $k = 2$ results in a less accurate model because the amount of training data per model is reduced. In other words, the transition probabilities in the Markov model are not well-constrained. This is confirmed in Table 3, where we increase the leg size from 4 locations to 12, and note that model error decreases as training data is added from more locations. Though larger leg sizes aid training, they may also reduce the usefulness of a leg's model, since the granularity is reduced.

Cluster	Gap Length Error	Run Length Error
1	0.001637	0.000237
2	0.000917	0.000241
3	0.001212	0.000223

Table 1: Model performance as a function of number of clusters, for a model order of 2. The 2-cluster case achieves the smallest gap-length error, while having a run-length error comparable to other cases.

Order	Gap Length Error	Run Length Error
0	0.013633	0.008994
1	0.001137	0.000597
2	0.000917	0.000241
3	0.001176	0.000457

Table 2: Model performance as a function of model order for the two-cluster case. The order-2 model achieves the smallest gap and run-length errors.

6.3. TCP Prediction Performance

We next conduct trace-based emulation to estimate TCP throughput using both model-generated synthetic traces, and real traces from the UAV flights. Specifically, we instrument two Linux machines to redirect a TCP flow under test to a user-mode tunnel program which 1) exerts a rate limit equal to that in the trace collection, and 2) forwards or drops packets based on the nearest packet reception event in the trace [7] [18]. The Linux machines ran the kernel version 2.6.24 and the default Linux TCP implementation [19] with SACK and timestamp options turned on, and using CUBIC congestion control. Each TCP flow under test lasted 150 seconds.

The raw packet traces had relatively poor average packet loss. As a result, emulated TCP flows experience frequent timeouts, leading to severe channel underutilization and sometimes indistinguishably low throughput numbers for traces with significantly different characteristics. In practice, link-layer retries are often used to improve loss conditions; we adopt this method in emulation and allow five retransmissions per packet to obtain reasonable TCP throughput numbers for comparison.

Leg Size	Gap Length Error	Run Length Error
12	0.000917	0.000241
6	0.001352	0.000824
4	0.004740	0.002296

Table 3: Model performance as a function of leg size. Gap- and run-length errors increase as the leg size decreases, due to the decreased size of the training set.

Trace Description	TCP Throughput (KB/s)	Validation Error
Validation trace	11.4	
Model $n = 1, k = 1$	13.7	20%
Model $n = 1, k = 2$	13.9	22%
Model $n = 2, k = 1$	10.4	9%
Model $n = 2, k = 2$	12.2	7%

Table 4: TCP peformance observed on several synthetic traces and a validation trace. The validation error is computed with respect to the validation trace. Each value is an average of 4 runs; in the case of validation traces, each run used a different subset of the validation data.

We performed the TCP emulation over a leg of the flight with good reception, i.e. few blocks with gaps spanning the entire block. We test models with orders $n = 1, 2$ and clusters $k = 1, 2$, as well as a validation trace, and present the TCP performance results in Table 4. We can see that the order $n = 1$ models underperform all order $n = 2$ models. For models of order $n = 2$, a $k = 2$ cluster model slightly outperforms the $k = 1$ cluster model. We observe that this improvement is small because the leg does not show strong clustered behavior, since it is chosen to have few long gaps in order to have sustained TCP throughput.

7. Future Work

We have evaluated the TCP performance prediction ability of a model over a simple circling UAV flight pattern. However, many applications might require testing under a large set of flight patterns. Therefore, an attractive future extension of this work would give us the ability to evaluate arbitrary flight paths by "stitching" them together from legs characterized by substitute models drawn from a small, experimentally-trained basis set. The main challenge lies in choosing the substitute models while minimizing the modeling error; potential criteria useful in the choice include mean distance, signal strength, or orientation of the UAV. A second challenge consists of choosing the experimental flight trajectory such that its leg models allow us to construct the largest possible set of stitched-together composite flight paths. Construction of such a basis set of models is likely to be time-consuming, but our modeling methodology presented in this paper can be easily adapted to new training data. In this light, our methodology can be viewed as a bootstrap process for subsequent online model training, an avenue of future investigation.

Use of UAV auto-pilot was invaluable in our experimental work because repeating the same trajectory let us collect a larger amount of training data per trajectory leg. Yet, experimental results based on autopilot control may be limited in their implications; environmental effects such as different wind speeds can lead to variations in repeated flights. In this paper we assumed that these variations are largely negligible. It would be prudent to test this assumption in the future, by examining how much variation our method can tolerate when combining the collected data.

8. Conclusions

By using a location-dependent runs-and-gaps model, we can predict TCP throughput over a varying ground-to-UAV wireless channel. This is achieved in spite of large channel variations due to dynamically changing conditions in communication distance, antenna angles, engine shadowing, etc., and TCP's sensitivity to distributions of packet loss (gaps) and delivery (runs) statistics. Two factors contribute to the success: (1) model training on clustered blocks of traces based on a relatively stable metric—mean gap-length, and (2) per-leg modeling, leveraging a stable cluster distribution in a local geographic neighborhood. This is a general methodology for modeling TCP behavior and can be applied to any UAV flight path.

Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under agreement numbers FA8750-09-2-0180 and FA8750-10-2-0180. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. Additionally, we gratefully thank Robert Gorman for his help with field logistics including the arduous task of mounting our UAV payloads.

References

[1] D. E. Comer, *Internetworking with TCP/IP: Principles, Protocols and Architectures.* Upper Saddle River, NJ: Prentice Hall, 2005.

[2] R. Caceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE Journal on Sel. Areas in Comm.*, vol. 13, no. 5, pp. 850–857, 1995.

[3] L. Ahumada, R. Feick, R. Valenzuela, and C. Morales, "Measurement and characterization of the temporal behavior of fixed wireless links," *IEEE Trans. Vehic. Tech.*, vol. 54, no. 6, pp. 1913–1922, 2005.

[4] D. Hague, H. T. Kung, and B. W. Suter, "Field experimentation of cots-based uav networking," in *MILCOM*, 2006.

[5] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std., 1999.

[6] G. Judd and P. Steenkiste, "A simple mechanism for capturing and replaying wireless channels," in *SIGCOMM Workshop on Exp. Approaches to Wireless Net Design and Anal. (E-WIND)*, 2005.

[7] B. D. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz, "Trace-based mobile network emulation," in *SIGCOMM*, 1997.

[8] E. Gilbert *et al.*, "Capacity of a burst-noise channel," *Bell Syst. Tech. J*, vol. 39, no. 9, pp. 1253–1265, 1960.

[9] B. D. Fritchman, "A binary channel characterization using partitioned markov chains," *Trans. on Info. Theory*, vol. 13, no. 2, pp. 221–227, April 1967.

[10] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Proc. Winter Sim. Conf.*, 1996.

[11] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A markov-based channel model algorithm for wireless networks," *Wireless Networks*, vol. 9, no. 3, pp. 189–199, May 2003.

[12] X. Lu, Y.-C. Chen, P. Lio, and D. Towsley, "Modeling mobility from military manet traces," in *ACITA*, 2008.

[13] Y.-E. Lu, F. Wicker, Y.-C. Chen, P. Lio, and D. Towsley, "On secure network structures in the lakehurst trace," in *ACITA*, 2008.

[14] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "Glomosim: A scalable network simulation environment," UCLA, Tech. Rep., 1999.

[15] "Opnet," http://www.opnet.com.

[16] "Qualnet," http://www.scalable-networks.com.

[17] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," 1966.

[18] M. Satyanarayanan and B. Noble, "The role of trace modulation in building mobile computing systems," in *HotOS*, 1997.

[19] P. Sarolahti and A. Kuznetsov, "Congestion Control in Linux TCP," in *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference*, 2002, pp. 49–62.