REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
NOVEMBER 2015	TECHNICAL PAPER (Post Print)	SEP2013 – SEP 2014

4. TITLE AND SUBTITLE

CONTEXT AWARE TCP FOR INTELLIGENCE, SURVEILLANCE AND RECONNAISSANCE MISSIONS ON AUTONOMOUS PLATFORMS

5a. CONTRACT NUMBER
NA

5b. GRANT NUMBER
NA

5c. PROGRAM ELEMENT NUMBER
62788F / 625315

6. AUTHOR(S)

Brendon Poland, Michael Muccio, and Daniel Hague

5d. PROJECT NUMBER
T2SA

5e. TASK NUMBER
IN

5f. WORK UNIT NUMBER
H1

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RITF
525 Brooks Road
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/Information Directorate
Rome Research Site/RITF
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2015-005

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA Case Number: 88ABW-2014-4340
DATE CLEARED: 12 SEP 2014

13. SUPPLEMENTARY NOTES
Proceedings Military Communications (MILCOM) Conference, Baltimore, MD, 6-8 Oct 2014. This is a work of the United States Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Transport Control Protocol (TCP) provides reliable data transfer for a variety of applications that can be found throughout current deployed environments. Tactical edge environments with constrained bandwidth and throughput require all that is available of network capacities, specifically those of deployed Remotely Piloted Aircraft (RPA). These RPAs are required to collect and disseminate large amounts of Intelligence Surveillance and Reconnaissance (ISR) data to command and control centers throughout a given theatre of operations. RPAs can provide an opportunity to use the repeatable nature of flight operations to record and predict the performance of TCP. This paper presents the optimization opportunity of TCP.

15. SUBJECT TERMS
Transport Control Protocol (TCP), Remotely Piloted Aircraft (RPA), Intelligence Surveillance and Reconnaissance (ISR)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**BRENDON POLAND**
U	U	U	UU	8	19b. TELEPHONE NUMBER (*Include area code*) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Context Aware TCP for Intelligence, Surveillance and Reconnaissance Missions on Autonomous Platforms

Brendon Poland, Michael Muccio, Daniel Hague

Air Force Research Laboratory, Information Directorate, Rome, NY 13440
Email: Brendon.Poland@us.af.mil

***Abstract*—Transport Control Protocol (TCP) provides reliable data transfer for a variety of applications that can be found throughout current deployed environments. Tactical edge environments with constrained bandwidth and throughput require all that is available of network capacities, specifically those of deployed Remotely Piloted Aircraft (RPA). These RPAs are required to collect and disseminate large amounts of Intelligence Surveillance and Reconnaissance (ISR) data to command and control centers throughout a given theatre of operations. RPAs can provide an opportunity to use the repeatable nature of flight operations to record and predict the performance of TCP. This paper will present the optimization opportunity of TCP sessions on RPAs operating ISR missions. Additionally possible techniques will be analyzed for achieving improved average throughput of asymmetric links, such as are seen in the context of RPA ISR missions.**

Keywords—RPA, UAV, SUAS, TCP, ISR, Prediction, Context Aware, Tactical Edge

I. INTRODUCTION

Technology acquisition strategy throughout the U.S. Department of Defense (DoD) continues to evolve to provide solutions to warfighter's problems as fast as possible. Along with this process comes the need for systems to comply with commonly accepted systems engineering practices of interoperability, so they can be fielded in a quick and orderly manner. One of the products of this requirement is that systems are encouraged to utilize widely accepted standards to allow for rapid determination of compatibility to other systems. One of those standards is the use of Internet Protocol (IP) as a network/information transfer interface and therefore TCP or User Datagram Protocol (UDP) as a transport mechanism (if required). This circumstance places a greater requirement for optimization of commercial standards (such as TCP) that are currently not of interest to the commercial world due to differing technology requirements.

Full Motion Video (FMV) requirements of the commercial world may be satisfied with a UDP stream, under certain circumstances the DoD uses TCP for more reliable connection that supports more robust target acquisition and continuity. In operational scenarios, Remotely Piloted Aircrafts (RPAs) are utilized for many different missions, including Counter Improvised Explosive Devices (CIED), Counter Indirect Fire (CIDF), Pattern of Life (PoL), Base Defense, Convoy Escort and others. In these scenarios small RPAs traverse repetitive waypoints or loiter at a single waypoint while intelligence officers collect and exploit data from sensors on these RPAs.

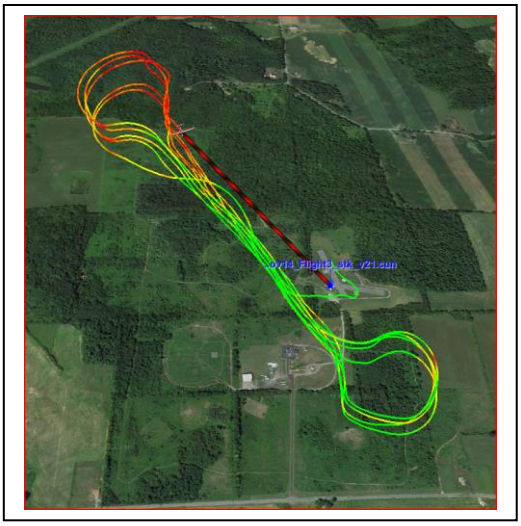

Figure 1: Change in SNR is shown by change of color of RPA flight path. Green having the highest SNR and red having the lowest.

It is in these scenarios, that communication connections are often interrupted due to the complex geography or obstacles in the route, which leads to the loss of packets and consequently a decrease in throughput. The implementation of a context-aware TCP scheme is expected to provide an increased throughput from data source to destination. Figure 1 shows the flight path of a small RPA in a PoL flight path scenario. The change of SNR magnitude experienced by the receiver is shown by the color of the path that the RPA traverses in its flight path. This representation of link quality shows the repetitive nature of both the flight path of an RPA on mission and the electromagnetic characteristics that its data payloads will likely experience. The relationship among location, RPA system attitude and link quality can be propagated up the network stack and be shown to have a direct impact on the TCP session that is reliably moving data from air to ground.

In this paper, a context-aware TCP protocol has been proposed to reduce the impacts of diverse electromagnetic environments (such as those seen by RPA comm links) on network throughput by leveraging the knowledge that RPAs fly in repeatable and therefore predictable flight paths. This paper will describe an algorithm that has been designed to observe,

predict and improve the TCP congestion back off mechanism to allow for an overall increase in throughput.

II. TCP Optimization potential

A. TCP and Congestion Avoidance

Reliable information transmission via TCP has been a crucial part of the development of the internet and other mainstream information transmission systems used to date. [1] One of the most commonly discussed topics within TCP is the congestion avoidance mechanism. This mechanism started as a conservative way to prevent destructive throughput oscillations and potential communication blackouts. Over time it has evolved to provide stable throughput throttling via additive increases and multiplicative decreases (to the congestion window size) for TCP sessions. [2]

B. TCP for wireless environments

With respect to wireless communication and infrastructure TCP has seen a variety of different improvements to deal with random errors, high latency and other complications. However, TCP still lacks the ability to accelerate and decelerate at the same rate as the electromagnetic medium that it is using to communicate. Evidence of this statement can be seen in Figure 2. While the electromagnetic medium (blue line) is healthy at the beginning of the test, the TCP session maintains a conservative growth rate when a more aggressive rate could be used. Additionally, near the end of the test when the medium is recovering from being submersed below the electromagnetic noise floor the window size limits the growth to much less than could actually be achieved.

C. TCP Reno

TCP Reno represents a very basic yet complete version of the congestion control algorithm and a very good benchmark [1]. Although other specialized versions of TCP (TCP cubic and Compound TCP) are currently being used in many Linux and Windows platforms they offer very minor changes and are tailored for more specific applications. TCP Reno is a good bench mark because it performs "good enough" on good to fair wireless links but backs off quickly once more than 1 or 2 packet losses occur. Figure 2 shows a bench test of TCP Reno performance over a couple minutes. The slow build up of the congestion window during times of consistent signal strength demonstrate opportunities to distinguish between true congestion and electromagnetic insufficiencies for data transmission.

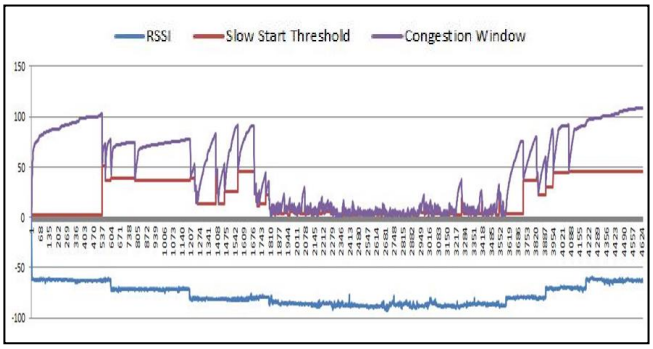

Figure 2: TCP Reno congestion window size (packets) is show in purple and demonstrates the characteristic additive increase and multiplicative decrease. RSSI (dBm) is shown in blue and slow start threshold (packets) is shown in red.

III. TCP Stockbridge

It is important to first identify why an improvement to TCP is possible and what must be done to accomplish it. In the simplest terms, it's a race between the transport layer and the physical layer and TCP has historically been losing. TCP Reno cannot vary its window size as fast as the wireless link quality can. Therefore, there is wasted time when the link quality is good enough to send more data but TCP cannot detect it fast enough to take advantage of it. This is where leveraging the knowledge that RPAs fly repetitive flight paths can help. If the TCP algorithm knows that the link is going to improve or that the link has already improved we can change the rules for how the window size works and capture this lost transmission time. This of course is not a new idea. It is important to note that some of the foundation for this work was accomplished and published in 2010 under an effort with Harvard University and the Office of the Secretary of Defense (OSD) [3]. Based on [3] and other RPA link characterization work done at the Air Force Research Laboratory (AFRL) owned Stockbridge Research Facility in upstate New York [4], [5], the ability to design, emulate [5] and flight test new algorithms has become very quick and affordable. Due to that history and the current capabilities of the test site that is being used for this work, the TCP modifications that will be examined in this section will be referred to in this document as TCP Stockbridge.

A. Rationale of Algorithm

TCP Stockbridge operates under several assumptions. The first is that RPAs operate in repetitive flight paths and the precision by which they navigate produces reliable metrics by which prediction can be accomplished. Figure 3 shows link metrics collected on an ISR style RPA flight accomplished in 2011. The second assumption is that TCP does not need to throttle back for random sparse errors that can be determined to be related to the nature of wireless links and not network congestion or bad link quality. This is done by using a heuristic that helps inform TCP of the health of several of the network layers. The result of this heuristic is the capability to inform TCP of current and future expected wireless performance.

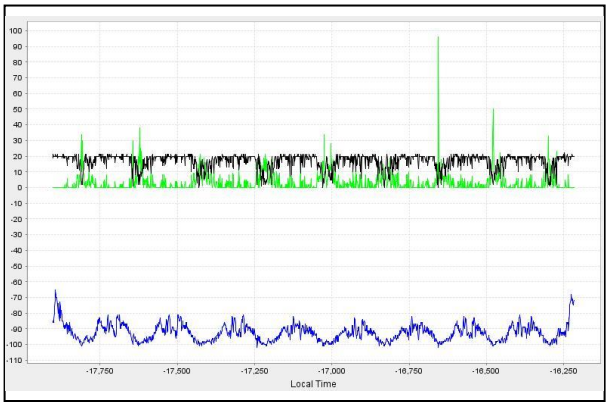

Figure 3: RPA link characteristics. Blue is SNR (dBm), Green is packet errors (packets/sec) and black is throughput (packets/sec)

This can be seen visually in Fig. 4 when compared to Fig. 2. Note that the blue on the bottom of both graphs is the SNR. Both TCP sessions were exposed to very similar electromagnetic characteristics and TCP Stockbridge was able to maintain a higher congestion window size despite areas of lost packets which caused TCP Reno to throttle back. The comparison between figure 2 and figure 4 shows the performance differences and how this improvement could enhance effective throughput. This scenario was also simulated using Matlab and past TCP error distribution data files from past flight tests. The outcome of the simulation along with bench testing showed a potential TCP rate increase of 2.5 times that of TCP Reno. This increase in rate was calculated in Maximum Segment Size (MSS) packets per second (due to greater window sizes). For both the bench test and the simulation the heuristic was employed to evaluate the local electromagnetic environment and make decisions on whether or not to modify the reaction to a congestion event. The heuristic uses a simple linear combination of several statistics about the link quality available on the RPA.

Figure 4: TCP Stockbridge congestion window size (packets) shown in Green, RSSI is shown in blue (dBm) and slow start threshold (packets) in shown in red.

B. Algorithm Description

The statistics used to generate the heuristic include SNR, retransmit timeout, round trip time, round trip time variance, time since last ACK received, time since last data sent, and ACKs per second. A bias and scale was applied to each factor and the resulting values were combined into a single value useful for decision making purposes. SNR and ACK's per second were weighted the highest as they are first order feedback on the link health, while the other statistics are mostly seen as confirming variables. Figure 5 shows SNR (RSSI), RTT, RTT variance, ACKs/second and the heuristic in yellow. This information is used in the algorithm to know when errors can be ignored (High SNR) and when errors need to be allowed to decrement the congestion window (Low SNR or bad current throughput). As the RPA flies its path for the first time, TCP Stockbridge collects the metrics described and makes decisions on 200ms intervals based on current (200ms old) data. Once the RPA begins passing over its original path it begins to use data from the "future" (200ms in the future) to decide how aggressively it will ignore the normal TCP back-off mechanism.

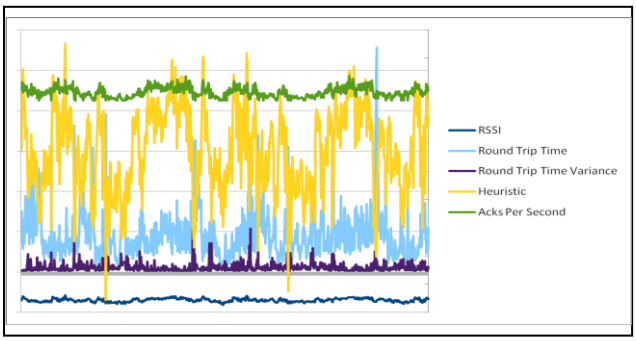

Figure 5: TCP Stockbridge congestion window size (packets) shown in Green, RSSI (dBm) is shown in blue, slow start threshold (packets) is shown in red and the heuristic (unitless) is shown in yellow.

C. TCP Stockbridge Psuedo Code

User Space pseudo code

```
list := vector[data]
lookup := rtree[gps,index]

every 200ms
  # get and store location, RSSI, and tcp info
  APPEND(list, READ_DATA())

  # smooth the noisy data
  if LENGTH(list) >= SMOOTHING then
    target := LENGTH(list) - SMOOTHING / 2
    list[target] = AVERAGE_LAST(list, SMOOTHING)

    # insert smoothed data into the rtree
    INSERT(lookup, list[target].GPS, target)
  end

  # find nearby smoothed samples
  results := QUERY(lookup, LAST(list).GPS)

  # attempt to time shift along the repeating orbit
```

```
  foreach result in results
    result += LOOK_AHEAD

  # find the average RSSI of the nearby points
  rssi := AVERAGE_INDEXED(list, results).RSSI

  if rssi < RSSI_LOW_THRESHOLD then
    # default to Reno
    SET_RF_QUALITY(BAD)
  else if rssi < RSSI_MODERATE_THRESHOLD then
    # forgive some packet losses
    SET_RF_QUALITY(POOR)
  else if rssi < RSSI_GOOD_THRESHOLD then
    # forgive a few packet losses
    SET_RF_QUALITY(FAIR)
  else
    # open the congestion window to discovered maximum
    SET_RF_QUALITY(GOOD)
  end
end
```

Kernel Space pseudo code

```
initial_window <- /proc/stockbridge/initial_window
rf_quality <- /proc/stockbridge/rf_quality

on new connection
  SET_CONGESTION_WINDOW(initial_window)
end

on congestion event
  # there is no way to avoid this happening in the Linux kernel
  SET_CONGESTION_WINDOW(GET_CONGETSION_WINDOW() / 2)

  # change behavior based the RF quality
  forgive := 0

  switch(rf_quality)
    case(BAD)
      # do nothing
    end

    case(POOR)
      forgive = FORGIVE_WHILE_POOR
    end

    case(FAIR)
      forgive = FORGIVE_WHILE_FAIR
    end

    case(GOOD)
      forgive = FORGIVE_WHILE_GOOD
    end
  end

  # only additively decrease if the losses are spread out temporally
  if CONSECUTIVE_LOSSES() <= forgive then

SET_CONGESTION_WINDOW(PREVIOUS_CONGESTION_WINDOW()
- 1)
  end
end
```

IV. Experimental Study

A. *Initial Test Setup*

Once the first version of the algorithm (pseudo code seen in section III.C) was finalized it was installed on an Artigo standalone computer. Attached to it was a Wi-Fi adapter running in infrastructure mode. A Linksys WRT54G AP was setup as the destination. To make sure the link was being exercised to its fullest, the source's (on the RPA) buffer was being kept 95% full so there was no chance of it not having anything to transfer. The ability of the wireless interface is of course limited by the media access control (MAC) mechanism for Wi-Fi, which reduces the signaling rate depending on RSSI. CSMA/CA (medium collision avoidance) was also running but due to the rural nature of the test location no other wireless networks were competing for spectrum. This payload was ground tested on a small UTV with varying distances and terrain.

During preliminary ground tests both TCP Reno and TCP Stockbridge experienced timeouts due to high latency conditions of long distances and mobile Wi-Fi clients. The standard Wi-Fi function "TCP_Low_Latency" was turned on to allow for more time before TCP timeout would be triggered. Once this option was turned on both links were relatively reliable. A Linux computer attached to the AP was set up to receive the stream from the source (on the aircraft) and log data. TCP dump was also implemented as an additional method for data logging. During the testing there was no control of the airborne payload as not to interfere with the data being sent through the network. Upon landing SSH was used to transfer log files from the source and data graphs were generated to show performance.

B. *Flight Operations/Facilities*

The data shown in this paper was collected over the course of several weeks at the Stockbridge Research facility located just south of Oneida, New York (Fig. 6). The Facility is a satellite site of the Air Force Research Laboratory's Information Directorate which is located in Rome, NY.

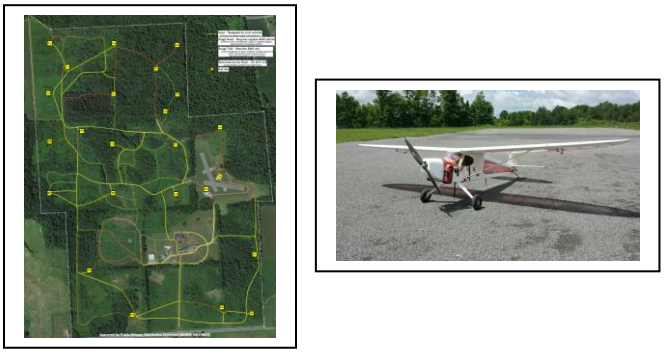

Figure 6: Stockbridge Research Facility on left and test aircraft on right.

The flight tests done for this effort were accomplished under the Unmanned Vehicle Experimental Communications Testbed (UVECT) flight test plan and were done over the Stockbridge Research Facility in the National Air Space. The

test site consists of 300 acres with 30 distributed ground nodes connected by terrestrial fiber and power. The site also has 2 600 foot runways for small RPA operations. The flight path was controlled by the onboard autopilot with an on hand flight crew of 3.

C. Initial results

Once Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) testing was complete to make sure the payload did not interfere with the command and control systems of the aircraft several flight paths were selected to exert the link and the TCP session running on top of it. A flight path that had varying distances from the AP was constructed and can be seen in Fig 7. This flight path was designed to degrade the link to approximately 50% of its capacity in one direction and 25% in the other. It provided a great scenario for varying conditions pertinent to potential operational ISR scenarios.

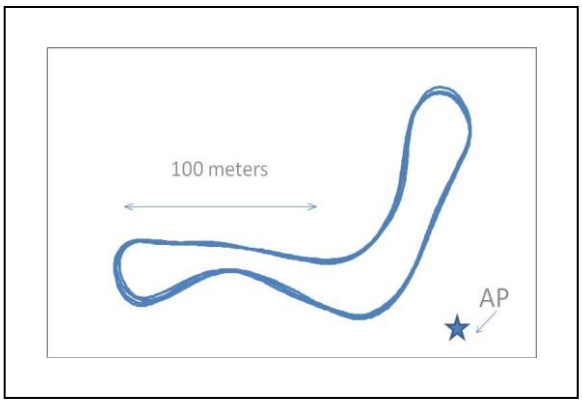

Figure 7: RPA flight path at 150 meters altitude intended to cause partial link loss to the north and almost complete loss to the west. There are 4 overlapping flight paths on this graph.

Baseline testing of TCP Reno was accomplished first and is shown in Fig 8. It is easy to see that the large peaks are the time the aircraft spent on the southeast corner of pattern, closest to the AP. The smaller peaks are the time the aircraft spent in the inside "elbow" of the pattern also relatively close to the AP. You can also see the repetitive nature of the TCP session in the transport layer as we assumed we would.

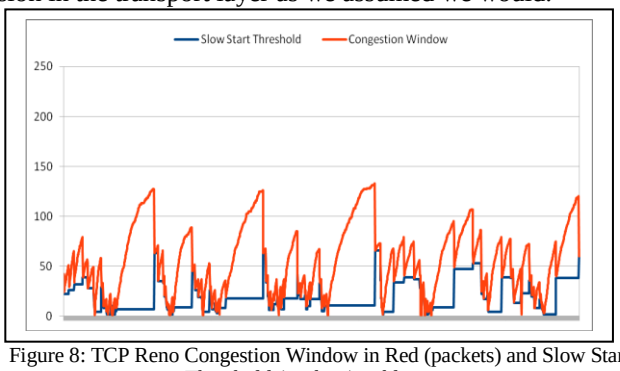

Figure 8: TCP Reno Congestion Window in Red (packets) and Slow Start Threshold (packets) in blue

Following the TCP Reno baseline collections, TCP Stockbridge was tested over the same flight path for the same number of laps. The results can be seen in Fig 9. Once again, the red line is the TCP window size and the blue line is the Slow Start Threshold (SST). The reason the SST is graphed with the window size is that each time the magnitude of the SST line changes value it means a perceived congestion event (Actually an RF event) was experienced and the slow start threshold was adjusted to half the current congestion window size. This helps us understand what events we were actually able to completely avoid through implementation of the new algorithm (change in Blue but no change in red) and which events were bad enough that we had to back off anyway due to no ACKs being received by the destination.

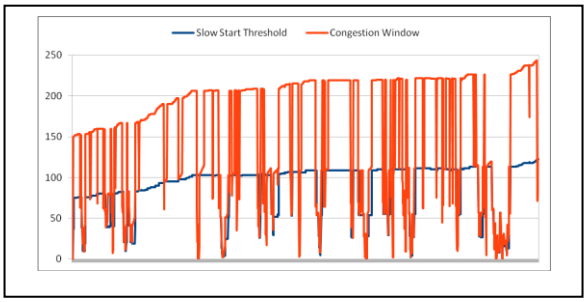

Figure 9: TCP Stockbridge Congestion Window (packets) in Red and Slow Start Threshold in blue

It is important to note a few success and failures with this data. First, if we simply calculate the area under the congestion window curve we observe an increase of 2.58 times the "opportunity" to send data. That is actually very close to what our analysis predicted it would be. However when we pulled our TCP dump data down we were shocked to see that with all that extra opportunity to send data we actually transferred less data during the course of the test than standard TCP Reno did. Figure 10 shows that TCP Reno transferred 875MBs over the course of the test and TCP Stockbridge transferred 825MBs. We of course began to analyze how this could be. We realized that when the link was failing under control of TCP Stockbridge the number of packets that were becoming lost was huge. This was causing a very large number of unacknowledged packets and therefore a large amount of back and forth (Once the link improved) for those packets to be finally acknowledged before we could begin to send new packets. This was causing goodput to be low despite the bursts of high throughput that were being provided. This caused us to rethink out heuristic metric and how we would implement this prediction algorithm.

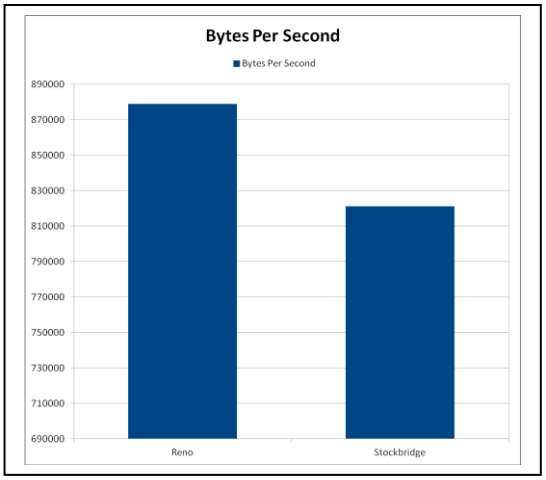

Figure 10: Average data rate for Reno and Stockbridge over the course of the whole test. Y Axis is mistakenly showing total Bytes for the test

D. Second Test Setup

Once we realized we were wasting all our opportunities to send more data due to the clean up we needed to accomplish, we decided to take our payload out of the aircraft and go back to the bench. We then took some of our TCP Reno baseline data and begin to analyze it for new opportunities to more effectively predict performance. We found that the SNR was incredibly volatile and was causing our heuristic to thrash up and down in magnitude. However the SNR was the key to knowing what kind of communication trend we were on. We therefore implemented a simple convolution or moving average to smooth the data. Starting with SNR data that was sampled several hundred times each 200ms, a Matlab moving average was implemented using a moving convolution function. Figure 11 shows a moving convolution of 75 200ms samples (That themselves contained hundreds of samples) and

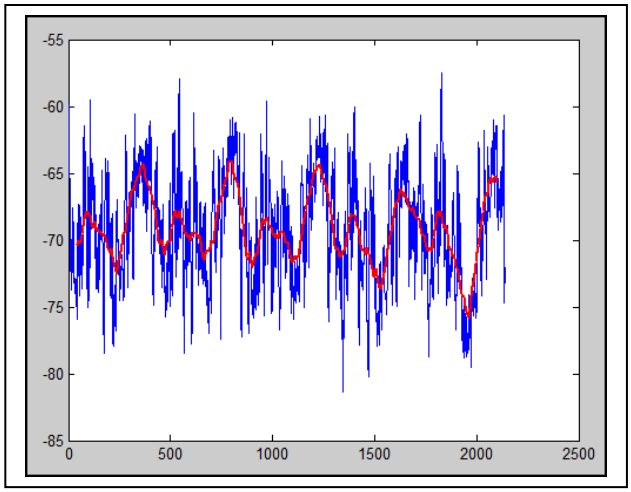

Figure 11: 200ms averaged SNR (dBm) in blue and 15second averaged SNR (dBm) in red

shows a very smooth function that shows great potential for our SNR trend line. If we then import the corresponding window size for this SNR data we can see the relationship between actual window size and a smoothed version of what the physical layer was experiencing. Figure 12 shows TCP Reno window sizes in blue and shifted and smoothed SNR in red from the last flight test we did.

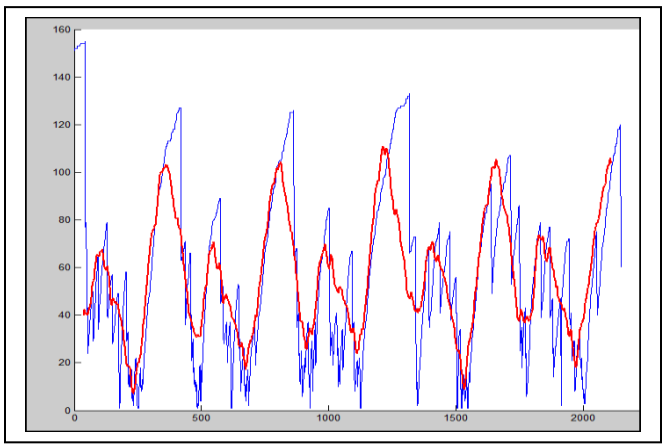

Figure 12: TCP Reno window size in Blue and SNR in Red

We changed the prediction algorithm to continue with the heuristic as a health metric but to trigger TCP Stockbridge behavior when upward trends (slope detection) in SNR are detected and a threshold SNR is reached. On the transition down (where we were very inefficient before) the heuristic reverts back to TCP Reno behavior just before the multiplicative decrease from the last pass. This will reduce outstanding unacknowledged packets and the ensuing overhead of cleaning them up. With these changes the payload was redeployed and another flight test was accomplished. Figure 13 is the TCP Reno baseline and Figure 14 is the TCP Stockbridge data from this test. Almost immediately it can be seen that TCP Stockbridge is much more stable. This change prevented the heuristic from thrashing the window size up and down but also unfortunately eliminated much of the increase in window size that was expected. This is due to the conservative and somewhat arbitrary initial thresholds that were put on the algorithm.

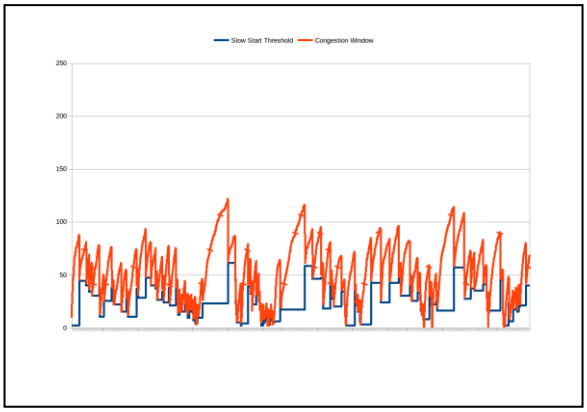

Figure 13: TCP Stockbridge Congestion Window in Red and Slow Start Threshold in blue

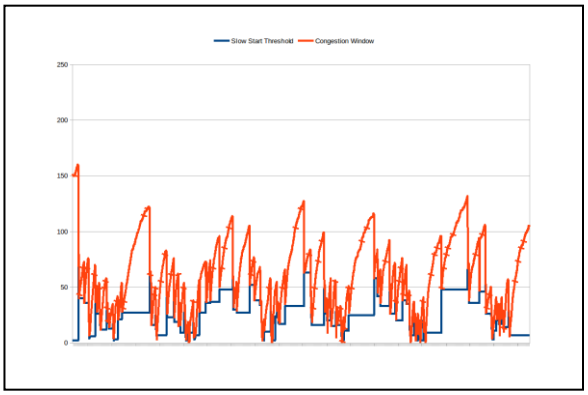

Figure 14: TCP Stockbridge Congestion Window in Red and Slow Start Threshold in blue

This being the case it is important to note that over the course of several tests that even with the very conservative slope detection thresholds in place (which eliminated almost all potential increases) TCP Stockbridge outperformed TCP Reno with respect to goodput as seen in figure 15.

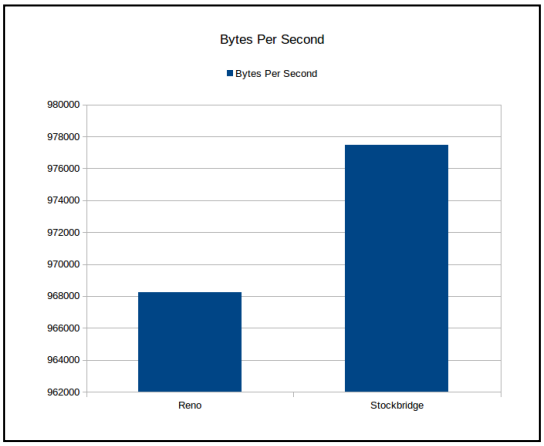

Figure 15: Average data rate for Reno and Stockbridge over the course of the whole test. Y Axis is mistakenly showing total Bytes for the test

It is assumed that with more development time the margin between TCP Reno and TCP Stockbridge will widen to a significant improvement.

V. Conclusions

This research area still has much to offer for continued network characterization and network improvements. Figure 9 and Figure 14 side by side show the potential for more improvements in throughput.

Future work will include optimizing the slope detection portion of the TCP Stockbridge algorithm and collecting more data from operationally relevant scenarios for TCP Stockbridge to be tested against. It will also be important to develop metrics to describe the variance and frequency of the physical layer for TCP calibration to different electromagnetic and network conditions. In conclusion, as RPAs continue to take on more and more missions, the DoD has much to benefit from an optimized version of TCP for these autonomous yet predictable platforms.

Acknowledgments: Thanks and appreciation goes out to the Stockbridge flight test support team, Robert Gorman, Jason Cassulis, Herb Bloss, Kevin Besig, and Mary Draper. Our flight test TAA, Dr. Michael Hayduk. AFRL/RI safety office: Michael Lovell, Michael Burke.

References

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

[2] James F. Kurose, Keith W. Ross, Computer Networking Third Edition, Pearson Addison Wesley 2005

[3] H. T. Kung, C.-K. Lin, T.-H. Lin, S. J. Tarsa, D. Vlah, D. Hague, M. Muccio, B. Poland, and B. Suter, “A location-dependent runs-and gaps model for predicting tcp performance over a uav wireless channel,”in MILCOM, 2010.

[3] 2008-2010 AFRL/RI Research Program “Characterization of the UAS Network Environment” (CUNE)

[4] 2010-2012 AFRL/RI Research Program “Protocol Emulation for Next Generation UAV Networks” (PENGWUN)

[5] Carvalho, Advances in Intelligence Modelling and Simulation. Springer 2012