Opportunistic Transport for Disrupted Airborne Networks

Justin P. Rohrer*
Department of Computer Science
Graduate School of Operational & Information Sciences
Naval Postgraduate School
Monterey, CA 93943–5285
jprohrer@nps.edu
http://faculty.nps.edu/jprohrer/

Kamakshi Sirisha Pathapati, Truc Anh N. Nguyen, and James P.G. Sterbenz
Department of Electrical Engineering & Computer Science
Information and Telecommunication Technology Center
The University of Kansas
Lawrence, KS 66045
{kamipks|annguyen|jpgs}@ittc.ku.edu
http://www.ittc.ku.edu/resilinets/

***Abstract*—Due to the challenging network conditions posed by a highly-dynamic airborne telemetry environment, it is essential for the transport protocol to provide automated mechanisms that dynamically adapt to changing end-to-end performance on any path. The AeroTP multi-mode transport protocol provides service tailored to the requirements of the telemetry mission control and data packets, achieving better performance compared to the traditional TCP and UDP. We use ns-3 to simulate the AeroTP protocol's reliable and quasi-reliable modes and demonstrate the performance tradeoffs between the modes, as well as comparing their performance with TCP and UDP.**

I. Introduction and Motivation

The highly dynamic airborne telemetry environment poses many unique challenging network conditions. One of the challenges is the constrained bandwidth caused by limited RF spectrum and the large quantity of data being sent over the network. The second challenge is the limited transmission range due to power and weight constraints of test articles (TAs). In addition, the high velocity of TAs poses the third challenge, the problem of mobility that results in highly dynamic topology changes. The fourth challenge is intermittent connectivity that arises due to the second and the third challenges. Unfortunately, the current TCP/IP-based Internet architecture is not appropriate for this environment [1] and there are several issues to be solved at the network and transport layers [2]. Given these constraints, in order to make the network resilient, we need to have cross-layer enabled protocols at the transport, network, and MAC layers that are uniquely designed to address the challenges posed by the aeronautical telemetry network. These protocols also have to be TCP/IP compatible with those devices located on the airborne nodes and with the control applications. With that in mind, in the context of the Integrated Network Enhanced Telemetry (iNET) program for Major Range and Test Facility Bases (MRTFB) across United States [3], we developed the Airborne Network Telemetry Protocol (ANTP) suite [4], [5]. The suite includes AeroTP – a TCP-friendly transport protocol that supports multiple reliability modes, AeroNP – an IP-compatible network protocol, and AeroRP – a routing protocol that takes advantage of location information to mitigate the short contact durations between high-velocity nodes. The different modes of AeroTP include reliable, nearly-reliable, quasi-reliable, unreliable connection, and unreliable datagram modes [6].

The AeroTP transport protocol in our ANTP suite was introduced in [6] and further developed in [5]. This paper extends the simulation and evaluation of its different modes and demonstrates the performance tradeoffs of each mode as well as comparing their performance with TCP and UDP. The rest of this paper is organized as follows: Section II presents some background and related work. Section III gives a detailed description of AeroTP simulation model. This is followed by our simulation results and an analysis on the performance of different modes in comparison to each other as well as to TCP in Section IV. Finally, Section V concludes the paper with directions for future work.

II. Background and Related Work

Ideally, reliable data transfer transmits data end-to-end with low delay and with no errors or losses. But, transmission in a network is often prone to delay, limited bandwidth, and multiple errors along the path towards the destination. Bit errors are the most common in wireless channels because of the channel's vulnerability to noise and interference. Packet losses are caused because of congestion, switching between multiple paths within the network, and packet-drops resulting from the occurrence of bit errors in the packet. To avoid the errors caused by congestion, congestion control and avoidance algorithms are used. When implicit congestion notification is used they reduce the window size each time congestion is detected. Packet drops at the receiver may be caused because of corrupted packets. Error recovery schemes are often a solution to correct the errors in the received data packet. The Automatic Repeat Request (ARQ) mechanism uses ACKs and retransmissions to ensure all the lost packets are successfully delivered to the destination. Forward Error Correction (FEC) is an alternative error-control mechanism that sends redundant

* Work performed while at The University of Kansas

information in each packet, allowing it to correct bit errors at the receiver without requesting a retransmission from the source.

A. Drawbacks of Traditional Protocols

Although TCP and UDP are the most commonly used transport protocols they fail to perform efficiently in a challenged wireless environment. In wireless networks packet losses are inevitable; link outages, lossy channel characteristics, unstable connectivity, delay, and congestion are a few examples of challenges that cause packet loss. A wireless channel is often subjected to interference and channel fading, resulting in packet loss and packet corruption. TCP assumes every packet loss is caused by congestion in the network and invokes its congestion control algorithm. This decreases the congestion window by a fraction (usually half), thus causing inefficient use of bandwidth. Schemes such as split-TCP connections and local retransmissions were developed to circumvent the problem caused by TCP's assumption of congestion being the only cause for packet loss [7].

TCP uses ACKs to provide reliable data transmission and retransmissions. The source retransmits a TCP segment to the destination when a timeout occurs while waiting for an ACK. A connection setup is performed through a three-way handshake between the source and the destination pair of nodes. This takes one round-trip time (RTT) before data may be sent, which causes significant performance degradation in a telemetry network because of short contact duration between nodes. By using a slow start algorithm, TCP takes many RTTs to ramp up the sending rate before it can fully utilize the available bandwidth. This results in a significant amount of wasted capacity in an environment that often has episodic connectivity.

Because of dynamic topologies, link outages are common. The congestion control algorithm is invoked during short link outages, causing an increase in the number of retransmissions. The connection is terminated in case of longer link outages. This causes difficulty in restoring links and finding alternate paths to the destination [4]. TCP also does not provide any QoS differentiation for prioritizing the type of data being transmitted.

SCPS-TP (Space Communications Protocol Standards – Transport Protocol) [8] is an extension to TCP, used particularly for satellite communications, developed to address problems posed by asymmetric links. SCPS-TP addresses some similar problems to those of telemetry networks although it is not fully suitable for highly dynamic networks. This is in part because it relies on channel condition information which is either pre-configured or discovered over time from the network [9].

Although UDP is a simpler protocol than TCP, it does not offer any guarantee for data delivery, so it is unreliable. Unlike TCP, UDP does not have a connection set-up mechanism and does not provide congestion or flow control or data retransmissions. UDP also does not provide differentiated

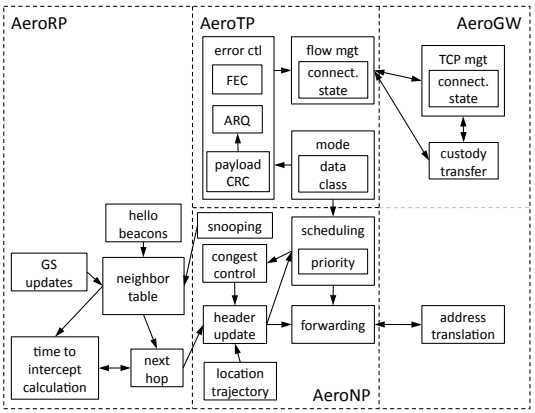

Fig. 1: Airborne network protocol functional block diagram

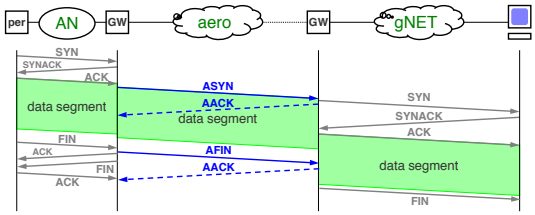

Fig. 2: AeroTP connection setup

levels of precedence or QoS for the classes of information required in the telemetry environment.

III. AeroTP: TCP-Friendly Transport Protocol

AeroTP is a new domain-specific transport protocol designed to meet the needs of the highly-dynamic network environment while being *TCP-friendly* to allow efficient splicing with conventional TCP at the AeroGWs in the GS and on the AN. Thus it transports TCP and UDP through the tactical network, but in an efficient manner that meets the needs of this environment: disruption tolerance, dynamic resource sharing, QoS support for fairness and precedence, real-time data service, and bidirectional communication. Table I identifies a number of key features of AeroTP and compares it to other modern and traditional transport protocols. AeroTP has several operational modes that support different service classes: reliable, nearly-reliable, quasi-reliable, best-effort connections, and best-effort datagrams. The first of these is fully TCP compatible, the last fully UDP compatible, and the others TCP-friendly with reliability semantics matching the needs of the mission and capabilities of the airborne network. The AeroTP header is designed to permit efficient translation between TCP/UDP and AeroTP at the gateway as described in Section III-B.

AeroTP performs end-to-end data transfer between the edges of the airborne network and either terminates at native Aero devices or splices to TCP/UDP flows at the AeroGWs. Transport-layer functions that must be performed by AeroTP include connection setup and management, transmission control, and error control, shown in Figure 1.

TABLE I: Feature Comparison of AeroTP, TP++, UDP, and TCP Variants

Feature	AeroTP (ResTP)	TP++	UDP	TCP NewReno	(CU)BIC -TCP	T/TCP	SCPS-TP
TCP Compatible	friendly	no	no	yes	yes	yes	interop
UDP Compatible	friendly	no	yes	no	no	no	no
3-way handshake	no	no	no	per-flow	per-flow	per-endpoint	per-endpoint
partial-path support	yes	no	yes	no	no	no	no
header integrity check	CRC-16	chksum	no	no	no	no	no
data integrity check	CRC-32	chksum	16-bit chksum	16-bit chksum	16-bit chksum	16-bit chksum	16-bit chksum
error correction	variable FEC	FEC	no	no	no	no	no
aggregated ACKs	yes	yes	no	optional	optional	no	yes
selective repeat	yes	yes	no	optional	optional	no	yes
negative ACKs	optional	no	no	no	no	no	optional
multipath friendly	yes	yes	no	no	no	no	no
flow control	x-layer	out-of-band signals	no	windowed	windowed	windowed	windowed
congestion ctrl	x-layer AeroNP backpressure	none	none	slow-start, AIMD, fast rexmit	slow-start, (CU)BIC, fast rexmit	estimate, AIMD	estimate, Vegas, fast rexmit
error control	hybrid, modular, adaptive,	hybrid, modular	none	ARQ	ARQ	ARQ	ARQ
reliability modes	reliable nearly-reliable quasi-reliable best-effort	reliable quasi-reliable	best-effort	reliable	reliable	reliable	reliable

A. Connection Management and Transmission Control

AeroTP uses connection management paradigms suited to the wireless network environment. An alternative to the overhead of the three-way handshake is an opportunistic connection establishment in which data can begin to flow with the ASYN (AeroSYN) setup message (shown in Figure 2). The flow of data is originated by a peripheral sensor (per) as a standard TCP session, translated into an AeroTP session by the gateway to traverse the airborne network, and then translated back into a standard TCP session by the gateway on the ground. The TPDU (transport protocol data unit) size may be discovered using the standard path MTU discovery mechanism [10], however given the specialized nature of these networks it is expected that the best performance will be achieved by setting the peripherals to use an appropriate MTU as determined by the slot size of the underlying TDMA MAC [11]. Closed-loop window-based flow and congestion control with slow start is not appropriate to the highly-dynamic nature of this network, therefore we use an open-loop *rate-based* transmission control with instrumentation from the network layer and determine an initial rate, with backpressure to control congestion, as described in [4], [12], [13] for AeroNP. Error control is fully decoupled from rate control [14], [15], and is service specific as described below.

B. Segment Structure and Gateway Functionality

AeroTP is *TCP-friendly*, meaning it is designed to efficiently interoperate with TCP and UDP at the gateways. To support this, AeroGW functionality [16], [17] provides IP–AeroNP translation [12] and TCP/UDP–AeroTP splicing. A packet may pass through two gateways on its path from source to destination. The ingress gateway converts the TCP segments to AeroTPDUs, while the egress gateway converts AeroTPDUs to TCP segments. It should be noted that ingress and egress gateways are not additional network elements in the tactical environment, but rather the gateway functionality will be built into ANs and GSs.

An AeroTP segment (shown in Figure 3) is structured for a bandwidth-constrained network so it does not encapsulate the entire TCP/UDP and IP headers, but is capable of converting to the TCP/UDP format at the gateways. To satisfy the end-to-end semantics it keeps certain fields in common with the TCP/UDP headers such as the source-destination port numbers, TCP flags, and the timestamp. The sequence number uniquely identifies AeroTP segments for reordering them at the receiving edge and for error-control purposes. The HEC (header error check) field performs a strong CRC on the header to detect bit errors caused by wireless channel, thus making sure the packet is correctly transmitted to the destination. In case the payload gets corrupted, AeroTP performs FEC on the payload. The payload CRC is used in the absence of a link-layer frame CRC and enables the measurement of bit-error-rate for error correction depending on the transfer mode used.

C. Protocol Operation

As a connection-oriented protocol, it is essential to define and maintain consistent states at the sender and the receiver to establish a connection for data transfer. The states either remain the same or evolve to another depending on the events and actions that happen within the protocol during a communication session. Figure 5 shows the AeroTP reliable mode packet flow-diagram, in which S is the source, D is the destination, and TmNS represents the telemetry network and Figure 6 shows the state transition diagram.

Similar to TCP, the AeroTP source-destination pair uses control messages (ASYN, ASYNACK, AFIN, and AFINACK)

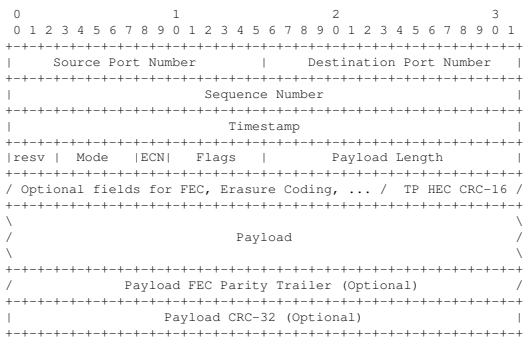

Fig. 3: AeroTP data segment structure

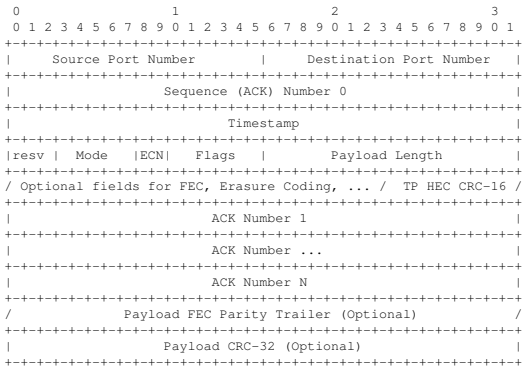

Fig. 4: AeroTP MACK segment structure

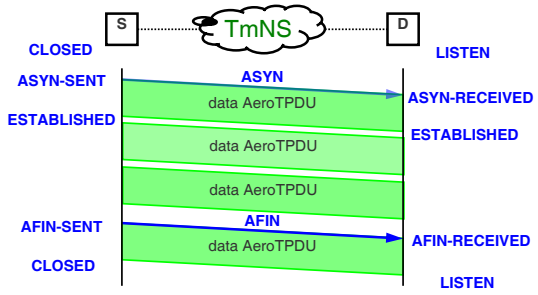

Fig. 5: AeroTP connection management

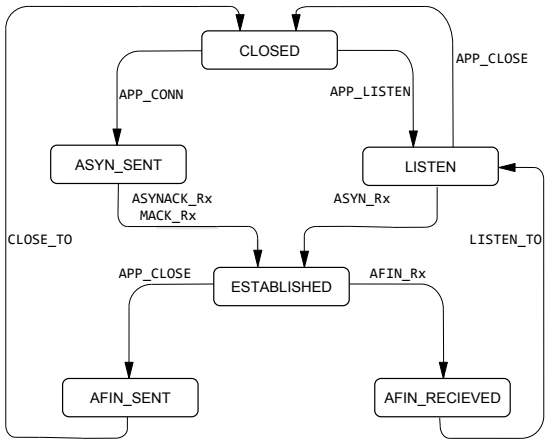

Fig. 6: AeroTP state transition diagram

TABLE II: State & Transition Definitions

State	Description
CLOSED	No connection & no data is transferred
LISTEN	Receiver is ready to listen to any incoming data
ASYN_SENT	ASYN message sent by the host initiating connection
ESTABLISHED	A steady state in which data transfer takes places
AFIN_SENT	AFIN message sent to indicate no new data being sent
AFIN_RECEIVED	AFIN message received
APP_CONN	Connection request issued by application
APP_LISTEN	Listen request issued by application
APP_CLOSE	Close request issued by application
ASYN_RX	ASYN received
ASYNACK_RX	ASYNACK received, connection est.
MACK_RX	Single or multiple packet ACK received
AFIN_RX	AFIN received, indicating end of any new data
CLOSE_TO	A timeout before going to the CLOSED state
LISTEN_TO	A timeout before going to the LISTEN state

for opening and closing a connection. The difference is an opportunistic connection establishment, in which data and control overlap, is chosen over the TCP's three-way handshake, thus saving a round-trip-time that is otherwise wasted. Initially the sender and the receiver are in the CLOSED state. A connection is initiated by the sender through an APP_CONNECT message from the application. The sender then sets the ASYN bit in the AeroTP header and transmits it with or without data depending on the data being available in the send buffer and moves to the AFIN_SENT state. The receiver receives an APP_LISTEN request from the application and moves to the LISTEN state, and upon receiving the ASYN message it moves to the ESTABLISHED state, and acknowledges the ASYN by setting the ASYN bit and the MACK bit simultaneously. The sender moves to the ESTABLISHED state as long as the ASYNACK or a simple MACK is received, so that the connection does not have to terminated in case the ASYNACK gets lost. While in the ESTABLISHED state, the sender and the receiver exchange AeroTPDUs and ACKs. After the sender is finished with sending all the data, an AFIN message (with AFIN bit set in the header) is sent to the receiver and the sender moves to the AFIN_SENT state. Upon receiving the AFIN message the receiver moves to the AFIN_RCVD state and transmits an AFINACK. To make sure that the receiver has acknowledged all the data including retransmissions, the receiver begins a timer in AFIN_RCVD state which goes to a LISTEN state after a time long enough so that all the retransmissions have likely been received from the sender. The sender also maintains a close timer, which expires after a certain time interval that is long enough to likely receive acknowledgements for all data packets. This way the sender is guarantees delivery of all the data packets with high probability.

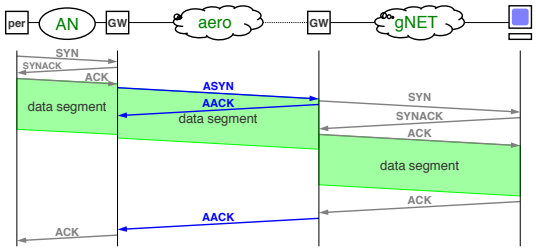

Fig. 7: AeroTP reliable connection transfer mode

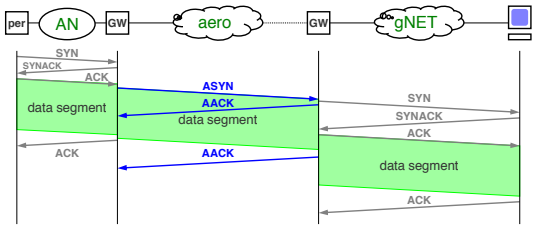

Fig. 8: AeroTP near-reliable transfer mode

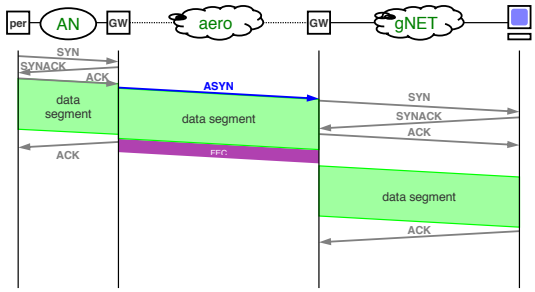

Fig. 9: AeroTP quasi-reliable transfer mode

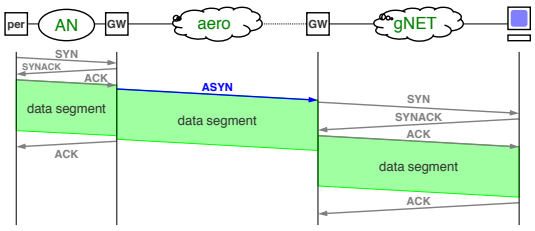

Fig. 10: AeroTP unreliable connection transfer mode

D. Error Control and QoS-Based Transfer Modes

Based on the application requirements, there will be a number a classes of data being transmitted over the tactical network. For this reason, AeroTP supports multiple transfer modes that are mapped to different traffic classes: reliable connection, near-reliable connection, quasi-reliable connection, unreliable connection, and unreliable datagram. These modes are slightly modified from the set found in ResTP to support the specific needs of the telemetry applications and environment.

All modes except unreliable datagram are connection-oriented for TCP-friendliness and use sequence numbers so that packets may follow varying or multiple paths and be reordered at the AeroTP receiver.

- ***Reliable connection*** mode (Figure 7) must preserve end-to-end acknowledgement semantics from source to destination as the only way to *guarantee* delivery. We do this using TCP ACK passthrough, which has the disadvantage of imposing TCP window and ACK timing onto the AeroTP realm, but will never falsely inform the source of successful delivery.
- ***Near-reliable connection*** mode (Figure 8) uses a custody transfer mechanism similar to that used in DTNs [18]–[20] to provide high reliability, but can not *guarantee* delivery since the gateway immediately returns TCP ACKs to the source on the assumption that AeroTPs reliable ARQ (automatic repeat request)-based delivery will succeed using SNACKs (selective negative acknowledgements) [8] supplemented by a limited number of (positive) ACKs as well as ELN (explicit loss notification) [21]. This still requires that the gateway buffer segments until acknowledged across the airborne network by AeroTP, but is more bandwidth-efficient than full source–destination reliability because TCP's ACK-clocked behavior only operates over the well-connected AN and ground-network (gNET) links, while allowing AeroTP to keep the assigned TDMA slots filled in the airborne network. However, the possibility exists of confirming delivery of data that the gateway cannot actually deliver to its final destination.
- ***Quasi-reliable connection*** mode (Figure 9) eliminates ACKs and ARQ entirely, using only open-loop error recovery mechanisms such as erasure coding, across multiple paths if available [22]. In this mode the strength of the coding can be tuned using cross-layer optimizations based on the quality of the wireless channel being traversed, available bandwidth, and the sensitivity of the data to loss. This mode provides an arbitrary level of statistical reliability but without absolute delivery guarantees.
- ***Unreliable connection*** mode (Figure 10) relies exclusively on the link layer (FEC or ARQ) to preserve data integrity and does not use any error correction mechanism at the transport layer. Cross-layering may be used to vary the strength of the link-layer FEC.
- ***Unreliable datagram*** mode (Figure 11) is intended to transparently pass UDP traffic, and no AeroTP connection state is established at all.

IV. Simulation Results

This section presents results from simulations of the AeroTP and AeroRP protocols performed using ns-3 and ns-2 respectively. The performance of AeroTP is compared to TCP, and the performance of AeroRP is compared to that of DSDV and AODV.

We have implemented ns-3 models of the fully-reliable (ARQ) and quasi-reliable (FEC) modes described in Section III-D. This section presents the simulation results from running these models. We compare the performance of AeroTP in the reliable connection and quasi-reliable modes with TCP

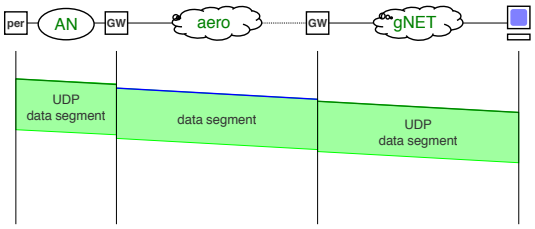

Fig. 11: AeroTP unreliable datagram transfer mode

and UDP protocols using the ns-3 open-source simulator [23]. The selective-repeat ARQ algorithm is used to provide reliable edge-to-edge connection between nodes for the reliable mode, and FEC (as discussed above) is used for the quasi-reliable mode of the AeroTP protocol. The network in this simulation setup consists of two nodes communicating via a link that is prone to losses. One node is configured as a traffic generator, and the other as a traffic sink. The traffic generator sends data at a constant data rate of 4.416 Mb/s (3000 packets/s with an MTU of 1500 B). The path consists of a 10 Gb/s link representing the LAN on the TA, a 5 Mb/s link with a latency of 10 s representing the mobile airborne network, and a second 10 Gb/s link representing the LAN at the ground station. Bit errors are introduced in the middle link with a fixed probability for each run, and the performance for each probability of bit-errors is shown in the plots described in the next section. A total of 1 MB of data is transmitted during one single simulation between the two nodes. The link is made unreliable by introducing losses using an error model varying bit-error probabilities ranging from 0 to 0.0001 for each of the protocols. Each simulation case is run 20 times and the results averaged to obtain the data needed for comparison.

A. AeroTP Connection Establishment

As mentioned previously, one of the drawbacks of TCP for highly-dynamic airborne environments is the three-way-handshake used for connection establishment. For this reason AeroTP is designed to establish a connection when the first data TPDU (with ASYN bit set) in a flow is received. If the first packet is lost, the connection can still be established using header information from the second or subsequent data packet, and the first packet can be retransmitted later if required by the specified reliability mode. To illustrate the difference between these two approaches, we have done simulations comparing the time required to establish a standard TCP connection, compared to a AeroTP connection.

The simulations are implemented in the ns-3 open-source simulator [23]. Each simulation consists of two nodes connected by a 10 Mb/s link with 5 ms latency and a fixed probability of packet loss, which is varied between 0 and 20% as seen on the x-axis. Node 0 is configured as a traffic generator (TCP or AeroTP as appropriate) and node 1 is configured as a traffic sink. For each packet-loss probability point plotted, the simulations were run 100 times and the results averaged. Each simulation consists of a single connection attempt by either TCP or AeroTP. We record the delay starting when the `connection_setup` command is issued to the transport protocol, and stopping when the first *data* packet is received by the data sink.

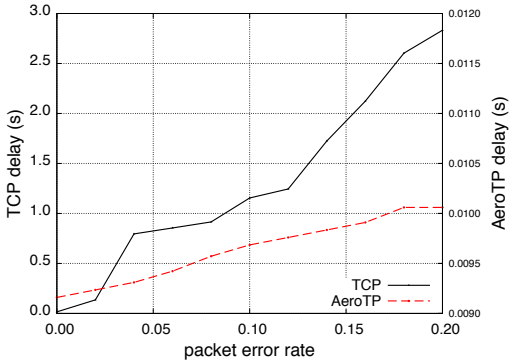

Fig. 12: TCP and AeroTP connection establishment delay

Figure 12 shows the results of these simulations. Both the TCP and AeroTP results are presented in a single plot, however, note that they are plotted against two different y-axes: TCP on the left, and AeroTP on the right. The TCP delay starts at about 20 ms when no losses occur, and increases linearly until it approaches 3 s when the packet-loss rate is 20%. AeroTP on the other hand, has a delay of 9.2 ms when no losses occur, and increases linearly to 10.1 ms when the packet-loss rate is 20%. This shows an improvement of two orders-of-magnitude, which will play a large role in enabling AeroTP to successfully send data over paths which only exist for a few seconds, while TCP would still be trying to establish the connection.

B. Fully-reliable mode performance

In *fully-reliable* mode, AeroTP uses ARQ as its reliability mechanism. This trades off additional latency (in the case of lost or corrupted packets) and overhead of the reverse channel, for reliability. The advantage to this mechanism is that given enough time, every lost packet can be retransmitted. In our model we are able to adjust the amount of bandwidth required by adjusting the number of ACKs aggregated into a single packet before it is transmitted. We found this to have a negligible effect on performance, and so have omitted the set of plots showing adjustments to this parameter to save space. The overall performance of the fully-reliable mode can be seen in our last set of plots.

C. Quasi-Reliable Mode Characterization

In *quasi-reliable* mode, AeroTP uses FEC as its reliability mechanism. This trades overhead on each packet for reliability. The advantage to this mechanism is low delay, because no retransmissions are required to correct errors. Our first set of plots compares varying FEC strengths, from zero FEC 32 bit words per packet, to 256 FEC words. In all cases 1500-byte packets are used, thus as the number of FEC words in each packet is increased, the number of bytes of data in each packet decreases, meaning that more packets

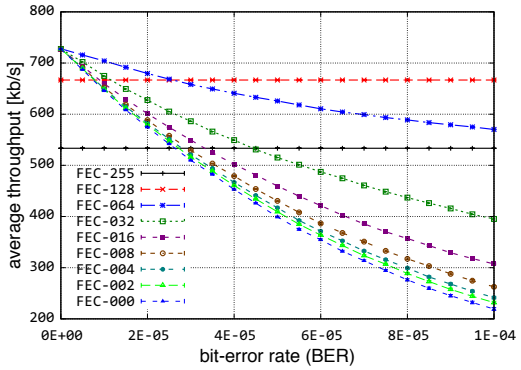

Fig. 13: Average goodput

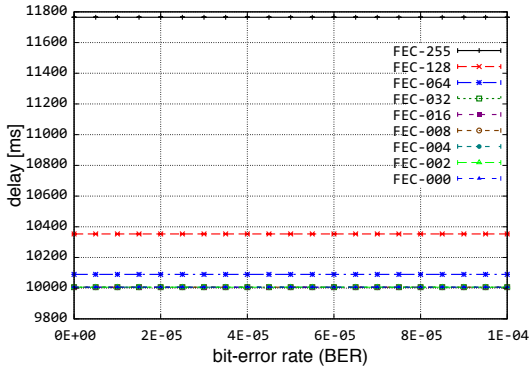

Fig. 14: Average delay

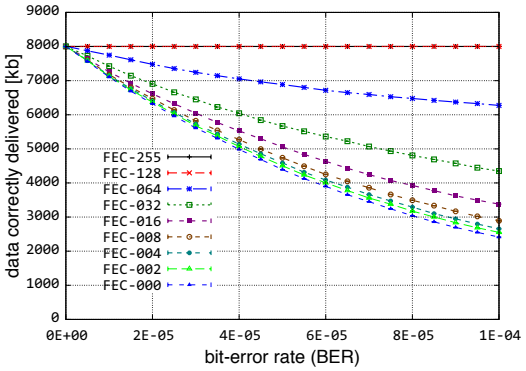

Fig. 15: Cumulative goodput

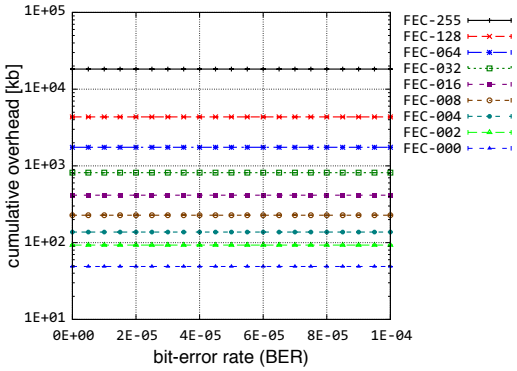

Fig. 16: Cumulative overhead

are required to transfer the same amount of data. Figure 13 shows that the throughput decreases due to uncorrectable packets as the error-rate increases, however this effect can be mitigated by increasing the FEC strength. For very high FEC strengths (128 and 256), there was virtually no decrease in performance across the range of error-rates tested, however the performance is decreased at low error-rates due to the high level of overhead. Due to the fact that retransmissions are not involved, the latency of data transmission is not affected by packet errors as shown in Figure 14. However, as very high levels of FEC result in link saturation this translates into added latency due to queuing delay. Similar to the throughput plot, Figure 15 shows the total amount of data transmitted. Depending on the FEC strength, this quantity decreases as the error-rate increases, except for very high FEC strengths (128 and 256) all errors are able to be corrected, at the rates tested. Lastly in this set of plots, we show the overhead imposed on the network by using the FEC mechanism at various strengths (Figure 16). This quantity is significant (note the log y-axis scale), however it is not affected by the error rate.

The next set of plots continue to characterize the *quasi-reliable* mode. Figure 17 shows that for error rates greater that zero, higher FEC strengths result in higher throughput up to a point. For the 128-word and 256-word FEC strengths, the amount of FEC bytes being sent begins to saturate the link, resulting in reduced throughput of data. Figure 18 shows that for all error rates, higher FEC strengths increase delay slightly as the link becomes saturated. Figure 19 shows that an FEC strength of 96 words/packet or greater is able to correct all errors at the error rates tested. Figure 20 shows the increase in overhead resulting from increased FEC strength. The increase is linear with respect to the amount of data being sent, but since we have chose to quantify FEC strength with respect to the number of packets transmitted it appears exponential, due to the fact that an increase in FEC strength results in an increase in the number of packets sent to transmit a give amount of data using maximum-size packets. Future models may change this quantification in favor of one relative to the amount of data being transmitted.

D. Performance Comparison over Lossy Links

Figure 21 shows that AeroTP reliable-mode is able to achieve significantly better performance than TCP, which backs off substantially as the BER (bit-error rate) increases. TCP also becomes highly unpredictable in its performance, as shown by the error bars. At the same time TCP's end-to-end delay increases by 3 orders-of-magnitude doubles with a BER of 1×10^{-4}, while AeroTP increases less than 1 order-of-magnitude as shown in Figure 22. Over the course of the simulation, both TCP and AeroTP are able to deliver the full 1 MB of data transmitted for low error rates $<$0.000035, but above that TCP performance drops rapidly while AeroTP is

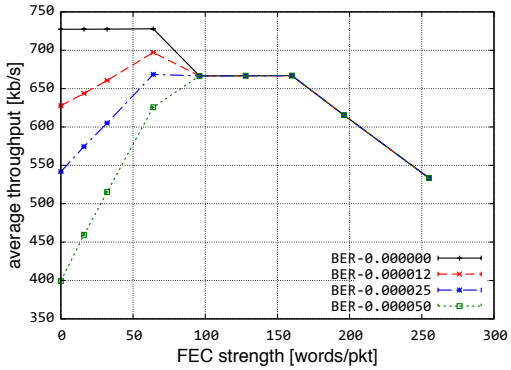

Fig. 17: Average goodput

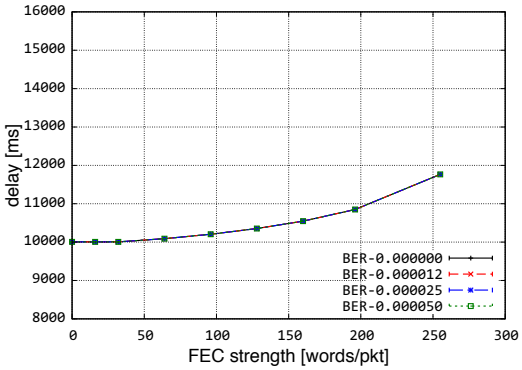

Fig. 18: Average delay

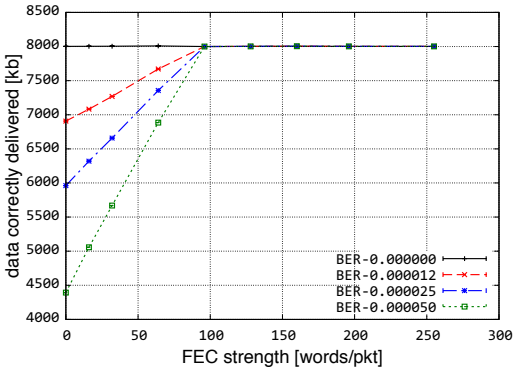

Fig. 19: Cumulative goodput

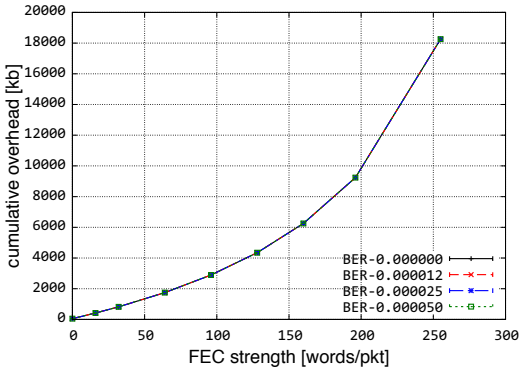

Fig. 20: Cumulative overhead

still able to deliver nearly all the data at the highest error rates as shown in Figure 23. In the same plot we see that UDP looses a percentage of the data due to corruption as the BER increases, and that the AeroTP quasi-reliable mode losses a much smaller percentage. Lastly in Figure 24 we see that the performance improvement of the AeroTP reliable-mode is achieved with much lower overhead than TCP, while quasi-reliable mode does incur significant overhead, but does not cause any increased delay as the BER increases.

V. Conclusions and Future Work

In this paper we presented the ns-3 simulation model of AeroTP, and results showing it's significant performance advantages in a lossy environment. We also show the tradeoffs enabled by having both a fully-reliable and a quasi-reliable mode. We are in the process of implementing this protocol to run in a linux environment along with the rest of the ANTP suite [24]. This will enable testing on mobile platforms as well as in real-world traffic conditions. In the future we will also be simulating and implementing gateways to allow translation between the traditional Internet protocol stack and the ANTP suite.

Acknowledgements

The authors would like to thank the Test Resource Management Center (TRMC) Test and Evaluation/Science and Technology (T&E/S&T) Program for their support. This work was funded in part by the T&E/S&T Program through the Army PEO STRI Contracting Office, contract number W900KK-09-C-0019 for AeroNP and AeroTP: Aeronautical Network and Transport Protocols for iNET (ANTP). The Executing Agent and Program Manager work out of the AFFTC. This work was also funded in part by the International Foundation for Telemetering (IFT). We would like to thank Kip Temple and the membership of the iNET working group for discussions that led to this work.

References

[1] "iNET Needs Discernment Report, version 1.0." Central Test and Evaluation Investment Program (CTEIP), May 2004.

[2] "iNET Technology Shortfalls Report, version 1.0." Central Test and Evaluation Investment Program (CTEIP), July 2004.

[3] "iNET System Architecuture, version 2007.1." Central Test and Evaluation Investment Program (CTEIP), July 2007.

[4] J. P. Rohrer, A. Jabbar, E. Perrins, and J. P. G. Sterbenz, "Cross-layer architectural framework for highly-mobile multihop airborne telemetry networks," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, (San Diego, CA, USA), pp. 1–9, November 2008.

[5] J. P. Rohrer, A. Jabbar, E. K. Çetinkaya, E. Perrins, and J. P. Sterbenz, "Highly-dynamic cross-layered aeronautical network architecture," *IEEE Transactions on Aerospace and Electronic Systems (TAES)*, vol. 47, pp. 2742–2765, October 2011.

[6] J. P. Rohrer, E. Perrins, and J. P. G. Sterbenz, "End-to-end disruption-tolerant transport protocol issues and design for airborne telemetry networks," in *Proceedings of the International Telemetering Conference*, (San Diego, CA), October 27–30 2008.

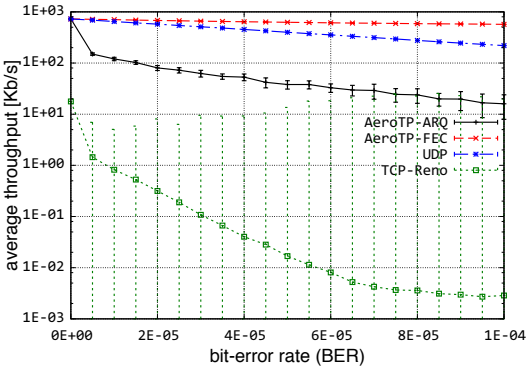

Fig. 21: Average goodput

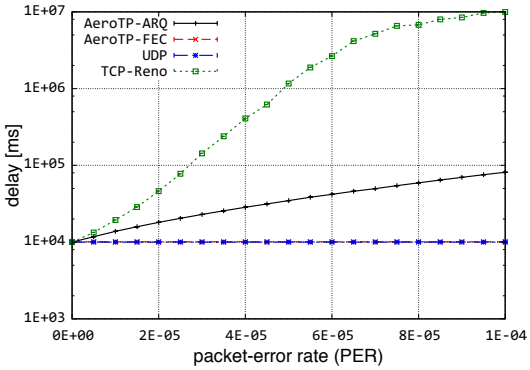

Fig. 22: Average delay

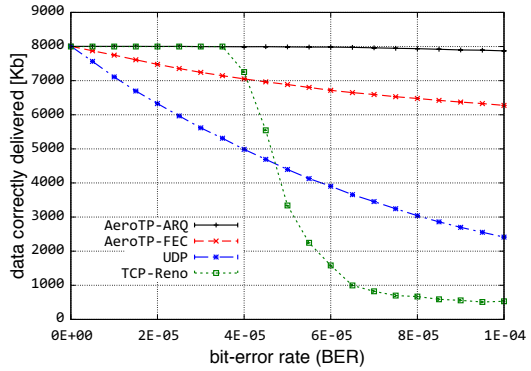

Fig. 23: Cumulative goodput

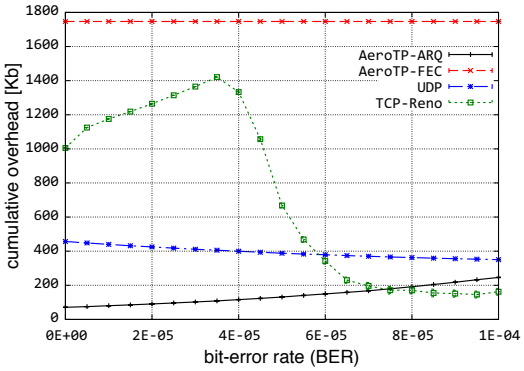

Fig. 24: Cumulative overhead

[7] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking (TON)*, vol. 5, no. 6, pp. 756–769, 1997.

[8] R. C. Durst, G. J. Miller, and E. J. Travis, "TCP extensions for space communications," in *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 15–26, ACM Press, November 1996.

[9] J. P. Rohrer and J. P. G. Sterbenz, "Performance and disruption tolerance of transport protocols for airborne telemetry networks," in *Proceedings of the International Telemetering Conference (ITC)*, (Las Vegas, NV), October 2009.

[10] J. Mogul and S. Deering, "Path MTU discovery." RFC 1191 (Draft Standard), Nov. 1990.

[11] iNET Working Group, "http://www.inetprogram.org."

[12] A. Jabbar, E. Perrins, and J. P. G. Sterbenz, "A cross-layered protocol architecture for highly-dynamic multihop airborne telemetry networks," in *Proceedings of the International Telemetering Conference (ITC)*, (San Diego, CA), October 27–30 2008.

[13] A. Jabbar and J. P. G. Sterbenz, "AeroRP: A geolocation assisted aeronautical routing protocol for highly dynamic telemetry environments," in *Proceedings of the International Telemetering Conference (ITC)*, (Las Vegas, NV), October 2009.

[14] J. P. Sterbenz and G. M. Parulkar, "AXON: Application-oriented lightweight transport protocol design," in *Proceedings of the Tenth International Conference on Computer Communication (ICCC)*, (New Delhi, India), pp. 379–387, Narosa Publishing House, November 1990.

[15] D. D. Clark, M. L. Lambert, and L. Zhang, "NETBLT: A high throughput transport protocol," *ACM SIGCOMM Computer Communication Review*, vol. 17, pp. 353–359, October/November 1987.

[16] "iNET TmNS Ground Segment Architecture, version 2007." Central Test and Evaluation Investment Program (CTEIP), July 2007.

[17] "iNET TmNS Test Article Segment Architecture, version 2007." Central Test and Evaluation Investment Program (CTEIP), July 2007.

[18] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, pp. 128–136, June 2003.

[19] K. Scott and S. Burleigh, "Bundle Protocol Specification." RFC 5050 (Experimental), Nov. 2007.

[20] K. Fall and S. Farrell, "DTN: An architectural retrospective," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 26, no. 5, pp. 828–836, 2008.

[21] R. Krishnan, J. P. G. Sterbenz, W. M. Eddy, C. Partridge, and M. Allman, "Explicit transport error notification (ETEN) for error-prone wireless and satellite networks," *Comput. Netw.*, vol. 46, no. 3, pp. 343–362, 2004.

[22] A. J. McAuley, "Reliable Broadband Communication Using a Burst Erasure Correcting Code," *SIGCOMM Comput. Commun. Rev.*, vol. 20, no. 4, pp. 297–306, 1990.

[23] "The ns-3 network simulator." http://www.nsnam.org, July 2009.

[24] M. AL-Enazi, S. A. Gogi, D. Zhang, E. K. Çetinkaya, J. P. Rohrer, and J. P. G. Sterbenz, "ANTP protocol suite software implementation architecture in python," in *International Telemetering Conference (ITC)*, (Las Vegas, NV), October 2011.