Comparison of Adaptive Transport Layer Error-Control Mechanisms for Highly-Dynamic Airborne Telemetry Networks

Kamakshi Sirisha Pathapati, Justin P. Rohrer
Faculty Advisor:
James P.G. Sterbenz
Department of Electrical Engineering & Computer Science
Information and Telecommunication Technology Center
The University of Kansas
Lawrence, KS 66045
{kamipks,rohrej,jpgs}@ittc.ku.edu

ABSTRACT

Transport protocols in highly-dynamic airborne networks call for adaptive error-control mechanisms to provide efficient error detection and recovery. Due to the highly dynamic nature of these networks and short contact durations between the nodes, the AeroTP protocol uses an adaptive multi-mode mechanism to provide a varying degree of reliable services to the application data. The two fundamental error-control mechanisms include an end-to-end ARQ mechanism to provide complete reliability in the AeroTP reliable mode and an end-to-end FEC mechanism to provide statistical reliability in the AeroTP quasi-reliable mode. In this paper, we present our implementation of the hybrid-ARQ mechanism in ns-3 to improve the throughput and delay performance of the AeroTP protocol. We also compare and analyze the performance of hybrid-ARQ against the different AeroTP modes, TCP, and UDP protocols.

I. INTRODUCTION AND MOTIVATION

The highly dynamic airborne telemetry environment is characterized by unique challenges that makes it difficult for traditional TCP/IP protocols to perform well. The challenges that are unique to this environment include constrained bandwidth caused by limited RF spectrum, limited transmission range due to power and weight constraints of airborne TAs (test articles), high velocity of TAs resulting in short contact durations between nodes, and mobility causing dynamic topology changes and intermittent connectivity [1]. These constraints demand domain-specific protocols that can adapt to the rapidly changing conditions. Hence, in the context of the iNET (Integrated Network Enhanced Telemetry) program, we developed the ANTP (Airborne Network Telemetry Protocol) suite [2, 3]. The ANTP suite of protocols are designed to address these challenges by employing cross-layering mechanisms between the transport, network, and MAC layers to operate efficiently. Although, the protocols developed are domain-specific in nature, they are also TCP/IP compatible as many of the iNET components and applications run on the current TCP/IP architecture [4]. 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.

AeroTP is a domain-specific TCP-friendly transport protocol designed to meet the needs of highly-dynamic airborne networks. AeroTP offers differentiated levels of precedence or QoS depending on the type of data being communicated. There are five operational modes: reliable connection, near-reliable connection, quasi-reliable connection, unreliable connection, and unreliable datagram [5]. Reliable connection preserves end-to-end acknowledgements to ensure a full guaranteed delivery using TCP ACK passthrough, near-reliable connection uses a custody transfer mechanism to provide high reliability but does not fully guarantee delivery, quasi-reliable connection provides statistical reliability using open-loop error recovery mechanism such as FEC (forward error correction) coding, unreliable connection relies on the link-layer (FEC or ARQ – automatic repeat request) to preserve data integrity but does not perform any error correction at the transport layer, and unreliable datagram passes UDP traffic with no AeroTP connection management involved [3]. The transport layer functions that must be performed by the AeroTP protocol include connection management, transmission control, and error control. Connection management involves setting up connections, and terminating them by synchronizing the sender and receiver using a defined state machine. Transmission control is rate-based with feedback from the network layer (AeroNP). The protocol primarily uses ARQ and FEC for error control and it is fully decoupled from rate control [2].

The AeroTP protocol was introduced and further developed previously as ns-3 simulation models [3, 6, 7, 5], evaluating and comparing the performance of AeroTP reliable and quasi-reliable modes against one another and against the TCP and UDP protocols. Previously, we simulated the operation of fully-reliable mode with single ACKs. In this paper, we show the fully-reliable mode operation while aggregating a certain number of ACKs. We refer to this aggregated ACK packet as multiple-ACK (MACK pronounced em-ack). In very low channel error conditions, aggregating ACKs can reduce overhead in the reverse channel significantly and improve goodput. This is particularly beneficial in a network with assymetric links. However, the advantage becomes more apparent in higher channel error conditions, in which the overhead of retransmitting packets for the loss of single ACKs becomes significantly large. This paper also presents the ns-3 simulation of hybrid error control mechanisms that make use of both ARQ and FEC codes to correct errors and provide reliability. We present two different ways of implementing the AeroTP hybrid mode. One of them provides an improved reliability over AeroTP-FEC with the use of NACKs (negative acknowledgements) and the other provides full reliability by using both ACK and NACKs. We compare these modes against the previously developed AeroTP reliable connection and quasi-reliable connection modes.

The rest of this paper is organized as follows: Section II presents background and related work discussing drawbacks of TCP and UDP protocols and the error control mechanisms adopted. Section III gives a detailed description of the AeroTP Hybrid-ARQ simulation model. This is followed by our simulation results and analysis of the performance of the error control mechanisms in comparison to each other as well as with TCP and UDP in Section IV. Finally, Section V concludes the paper with directions for future work.

II. BACKGROUND AND RELATED WORK

End-to-end transport layer functions include providing reliability, flow control, error control, connection management, and congestion control, depending on application requirements. Transport protocol design decisions are affected by the applications as well as the network environment in which they operate. Some of the examples of applications are client-server applications, peer-to-peer file sharing, and multimedia streaming. However, regardless of the application a transport protocol is designed for or the environment it is going to operate in, the goal is to achieve efficient communication. Efficiency in transport protocols is often measured in terms of packet exchange overhead that affects the overall delay and throughput, complexity of algorithms implemented, and the size of the state space [8].

Amongst the services provided by transport protocols, reliability guarantees the source correct delivery of data to the destination. Ideally, it is desirable to transmit data end-to-end without any delay and with no errors or losses. However, wireless channels are prone to errors due to their vulnerability to noise and interference, as well as losses due to congestion and switching between multiple paths within the network. Error control is adopted to ensure reliability: detecting and recovering from losses and corruption. Error control schemes such as ARQ and FEC codes have been used to provide reliability. While these mechanisms are simple to implement, achieving high system reliability is expensive because they involve state retention, redundant information, data copying, buffer management, and packet exchange overhead [8, 9].

A. Drawbacks of Traditional Protocols

Two of the earliest and the most commonly used protocols are TCP (transmission control protocol) and UDP (user datagram protocol) [10, 11]. Despite their popularity they are known to perform undesirably in challenged wireless environments. A wireless channel is prone to bit-errors, interference, and channel fading resulting in packet loss and corruption. TCP assumes every packet loss is caused by congestion in the network and invokes a congestion control algorithm [12]. This decreases the congestion window by a fraction each time causing inefficient use of bandwidth. TCP connection setup is performed through a three-way handshake between the source and the destination pair of nodes. This takes one extra RTT (round-trip time) causing significant performance degradation in a telemetry network because of the short contact duration between nodes. By using a slow start algorithm, TCP takes multiple RTTs to exit slow start before it can fully utilize the bandwidth. This causes a significant amount of overhead in an environment that often has episodic connectivity. TCP does not efficiently perform flow control in a network with asymmetric links, since it requires a highly reliable ACK stream for self-clocking. Because of the dynamic topology, link outages are common and TCP's 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 [2, 13]. Although UDP is a simpler protocol than TCP, it does not offer any reliability. Unlike TCP, UDP does not have a connection set-up mechanism and does not provide error control, congestion control, or flow control. Both TCP and UDP do not provide any QoS for prioritizing the type of data being transmitted.

B. *Error Control Schemes: ARQ, FEC, and Hybrid*

Early work on designing general-purpose protocols to provide end-to-end transport services was based on assumptions that no loss or corruption of packets would occur during an end-to-end transmission between hosts, and no missequenced packets or duplicate packets would arrive at the destination [8, 9]. However, in wireless networks these set of assumptions result in improper behavior. These networks are frequently prone to losses caused by channel characteristics, network topology, and traffic conditions.

Error control involves two phases: error detection and error recovery. Error detection involves identifying errors caused by corruption, loss, duplication, and missequencing of information. This is achieved through the use of checksums, FEC codes, and acknowledgements. Error recovery involves using retransmissions, FEC codes, or both to recover from the error. These mechanisms often come with drawbacks that affect the performance of the protocol. For example, while ARQ provides high reliability it results in low throughput in high BER conditions. Moreover, while recovering losses using acknowledgements and retransmissions, it is important to consider the overall delay caused due to retransmissions. Using a purely-ARQ based error control scheme results in higher delay in high BER conditions. An alternative to using ARQ is using FEC codes to correct transmission errors. FEC codes involve adding extra bits to each data segment that are then used to correct errors at the receiver. Although FEC does not significantly increase overall delay and maintains a fairly high throughput compared to ARQ in high BER conditions, adding additional bits to every data segment increases the packet overhead [14, 15]. Furthermore, using FEC codes alone provides only statistical reliability since it does not recover beyond the capability of the code, nor recover from losses or missing data segments. When errors go undetected the data segment is still delivered to the application and this makes employing a good FEC code expensive as it increases the complexity of the decoding system. Taking into account the drawbacks of using an error control based purely on ARQ or FEC, hybrid error control combines both ARQ and FEC to obtain better performance in moderate error conditions [16, 17, 18].

Hybrid error control schemes, commonly referred as hybrid-ARQ, involve FEC within an ARQ scheme. The idea is to use FEC error recovery capability to reduce the number of retransmissions. This addresses the performance drawback of ARQ in high BER conditions. In the case that the FEC is unable to recover any errors, the receiver requests retransmission using negative acknowledgements instead of delivering the corrupted data segment to the receiver. This improves the reliability of the scheme compared to using only using FEC codes [15]. There are two types of hybrid ARQ schemes: type-I and type-II [14, 16]. Type-I hybrid-ARQ scheme is an approach that is designed for simultaneous error detection and correction. When a received data segment is detected with errors, the redundant information within that segment attempts to correct them. Upon successfully correcting the errors the segment is delivered to the application. In case the correcting code is unable to correct the errors, the receiver discards the segment and requests a retransmission. The received retransmitted segment undergoes the same error check and decoding process. When unsuccessful, retransmission is requested and this repeats until the destination receives a segment that can be successfully decoded. Type-I schemes are suited for channels with a relatively constant error rate. When an efficient error correcting code is chosen, this approach can reduce the number of retransmissions significantly and improve the overall delay and performance. This scheme comes with drawbacks especially at the extremes of channel error conditions. During fairly low error rate the addition of extra bits increases the overhead, while during high error conditions the error code word used may not be sufficient

to correct all the errors, thereby increasing the numbers of retransmissions and reducing the throughput [14]. Type-II hybrid-ARQ schemes are suited for channels with varying error rates. These schemes often use adaptive hybrid-ARQ approach in which the extra bits are added depending on the channel conditions. The retransmission strategy involved in using this approach has varied from only retransmitting the extra bits to correct the corrupted original segment to retransmitting the entire segment [17, 19]. In this implementation we use type-I hybrid-ARQ scheme.

III. AeroTP HYBRID-ARQ OPERATION

In this section we describe the ns-3 simulation model of the AeroTP hybrid mode. The purpose of this model is to understand the operation of hybrid-ARQ mechanisms and compare its performance with other AeroTP modes to discuss the trade-offs.

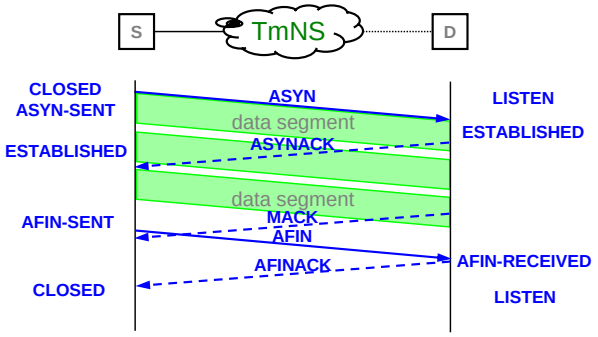

Figure 1: AeroTP connection management

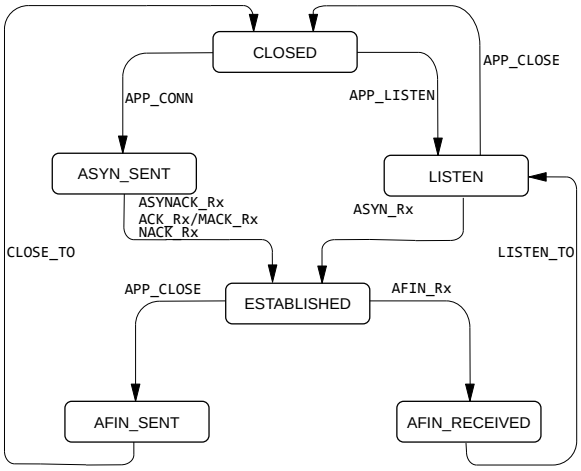

Figure 2: AeroTP state transition diagram

The AeroTP hybrid-ARQ operates in connection-oriented mode and thus shares the same state machine as that of the AeroTP reliable mode. Figure 1 shows the AeroTP reliable-mode packet flow-diagram, in which S is the source, D is the destination, and TmNS represents the telemetry network. Figure 2 shows the state transition diagram, which represents the sender and receiver states along with events that trigger the transition between them. Table 1 explains the function of all the states and events used to maintain a synchronized communication session between the source-destination pair.

While using either ARQ or FEC as the only error control mechanism is common, hybrid error control mechanisms can yield high end-to-end reliability [14, 15]. We implement two different hybrid error control mechanisms: AeroTP-NACK and AeroTP-NACK+MACK. The first mechanism AeroTP-NACK involves using NACKs to request retransmission from the sender when FEC fails to correct errors in the received TPDU (transport protocol data unit). This improves reliability when compared to the AeroTP quasi-reliable mode but does not offer full reliability as in the AeroTP reliable-connection mode. This is because NACKs are only sent for those TPDUs that are received but were unable to get corrected by the FEC code. TPDUs that may be lost during transmission are not recovered. The second mechanism AeroTP-NACK+MACK involves using both ACKs and NACKs. This approach uses ACKs to inform

Table 1: State transition definitions

State	Description
CLOSED	State in which no connection exists and no TPDU is transferred
LISTEN	State in which a destination is ready to listen to any incoming TPDU
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 TPDU being sent
AFIN_RECEIVED	AFIN message received and AFINACK is sent as an acknowledgement
Event	**Description**
APP_CONN	Request issued by the application to initiate connection by sending ASYN
APP_LISTEN	Request issued to the destination to move to the LISTEN state
APP_CLOSE	Request to intiate closing a connection by sending AFIN
ASYN_RX	ASYN received, indicating a connection has been requested
ASYNACK_RX	ASYNACK received, indicating connection request has been granted
ACK_RX	Single ACK received
MACK_RX	Multiple ACKs received
NACK_RX	NACK received indicating corrupted TPDU and requesting retransmission
AFIN_RX	AFIN received, indicating end of any new TPDU, initiating connection close
AFINNACK_RX	AFINACK received, indicating connection termination has been notified
CLOSE_TO	A timeout allowing outstanding retransmissions before going to CLOSED state
LISTEN_TO	A timeout to go to LISTEN state so that the receiver can receive all TPDUs

that data has been received correctly and NACKs to request retransmission of the original TPDU. This improves reliability over the first approach and is comparable in terms of reliability with the AeroTP reliable-connection mode because of the use of ACKs. Although this increases overhead when error rates and losses are low, it becomes advantageous in lossy conditions. It improves delay characteristics by reducing the number of overall retransmissions and requesting them only when an FEC code is unable to correct all the errors.

Figure 3 shows packet flow diagrams for AeroTP in reliable connection mode using only single ACKs, AeroTP in the hybrid-ARQ mode with only NACKs, and AeroTP in the hybrid-ARQ mode with both ACKs and NACKs. We can see that AeroTP in the reliable mode sends ACKs for every successfully received packet. The AeroTP operation in the other two modes is discussed in the following section.

C. AeroTP NACK Operation

As in AeroTP-reliable connection mode, the hybrid error control mechanism starts off with both the source and the destination in the CLOSED state. After receiving a connect message from the application, the source sends an ASYN message to the receiver, while moving to the ASYN_SENT state. Depending on the availability of the application data in the send buffer, the connection setup message is either an explicit ASYN message or an ASYN message piggybacked with the first TPDU. FEC words are added

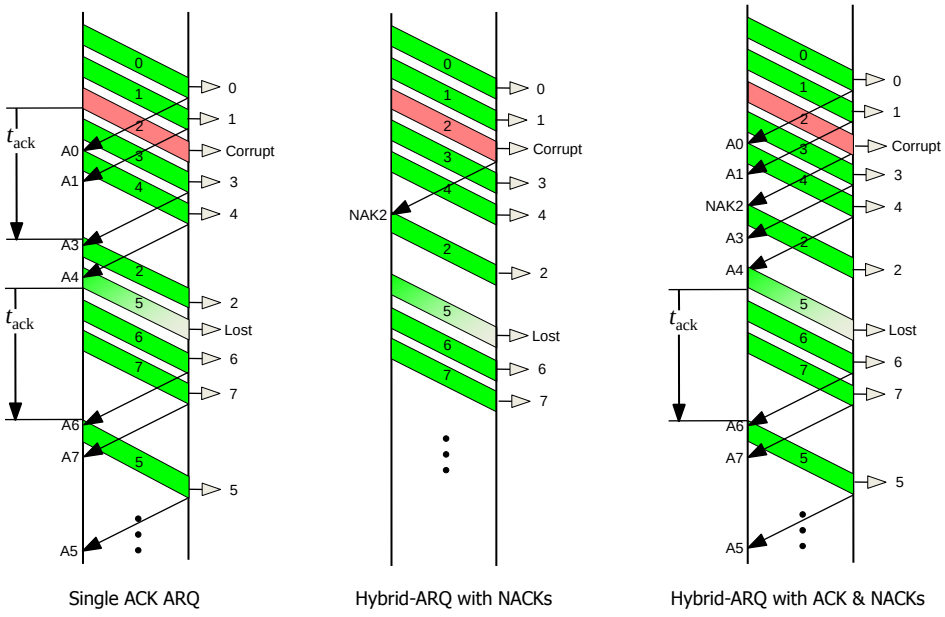

Figure 3: AeroTP packet flow diagram

to the payload depending on the FEC strength chosen. The destination receives a listen request from the application and moves to the LISTEN state. Upon receiving the ASYN message or AeroTP TPDU it moves to the ESTABLISHED state. The destination performs an FEC check on the ASYN message. If the FEC code is able to correct errors in the TPDU then it is delivered to the application, otherwise the destination sends a NACK to request retransmission of the original TPDU. The destination acknowledges the ASYN message from the sender and then it moves to the ESTABLISHED state. The source moves to the ESTABLISHED state when an ASYNACK is received. While in the ESTABLISHED state, the source and the destination exchange AeroTP TPDUs and NACKs. After the source finishes sending all the data, an AFIN message is sent to the destination while moving to the AFIN_SENT state. The destination moves to an AFIN_RCVD state and acknowledges the AFIN. To make sure that the destination has received all the TPDUs including retransmissions, the destination begins a timer in AFIN_RCVD state to go to the LISTEN state. The source maintains a close-timer, that expires after a certain time interval long enough to receive any request for retransmission with high probability. Furthermore, Figure 3 shows that lost TPDUs cannot be recovered in this mode.

D. AeroTP NACK+MACK Operation

The operation of the AeroTP NACK+MACK mode is similar to the AeroTP NACK mode. The difference lies with receiving ACKs or MACKs along with NACKs. For example, in the LISTEN and ESTABLISHED states upon receiving the ASYN message or AeroTP TPDU, the receiver performs an FEC check on the segment. If any errors are detected, the FEC attempts to correct them. When the receiver

successfully corrects the errors, it sends an ACK or MACK. When the receiver fails to correct the errors, the receiver sends a NACK requesting retransmission. As shown in Figure 3, when a segment is lost during transmission, the RTO (retransmission timeout) on the segment expires and the sender retransmits the data segment. Therefore all the data is delivered to the destination.

IV. SIMULATION RESULTS

We have discussed the operation of AeroTP hybrid-ARQ in Section III. This section presents the simulation results from running these models. We compare the performance of AeroTP hybrid-ARQ modes with AeroTP reliable connection and quasi-reliable modes, and with TCP and UDP protocols using the ns-3 open-source simulator [20]. AeroTP uses the selective-repeat ARQ algorithm to provide reliable edge-to-edge connection between nodes for the reliable mode, FEC for the quasi-reliable mode of the AeroTP protocol, and both in the hybrid modes. The network in this simulation setup consists of two nodes communicating via a lossy link. One node is configured as a traffic generator, sending data at a constant data rate of 4.416 Mb/s (3000 packets/s with an MTU of 1500 B), and the other as a traffic sink. The path consists of a 10 Gb/s link representing the LAN on the TA, a 5 Mb/s capacity 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 10^{-4} for each of the protocols. Each simulation case is run 25 times and the results averaged to obtain the data needed for transport layer comparisons with 95 percent confidence-interval bars plotted.

E. Fully-reliable Mode Performance

In the *fully-reliable* mode, AeroTP uses ARQ as its reliability mechanism. This trades additional latency (in the case of lost or corrupted packets) and overhead of the reverse channel, against reliability. The advantage of this mechanism is that given enough time, every lost packet can be correctly delivered to the application. The AeroTP reliable mode provides an option to aggregate multiple ACKs. With an MTU size of 1500 bytes we can aggregate up to 365 ACKs, in which each additional ACK add up to 4 bytes to the 20-byte AeroTP header.

The performance for different ACK aggregation values is measured in terms of average goodput, average delay, cumulative goodput, and cumulative overhead incurred as BER increases from 0 to 10^{-4}. Figure 4 shows decreasing goodput as BER increases due to fewer data packets delivered per unit time. Figure 5 shows that the average delay in general increases across the range of BER due to an increased number of retransmissions. However, there is no significant difference in the goodput or delay performance amongst different MACK values. We attribute this behavior to the amount of data transmitted being too small to see any significant difference in the performance. Figure 6 shows that all 1 MB of the data is delivered as expected, thus achieving full reliability. Figure 7 shows increased overhead for ACKs aggregated above 200 due to increased amount of retransmissions for every MACK lost.

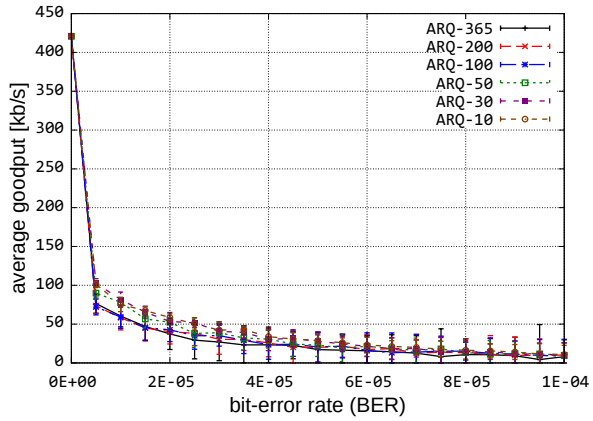

Figure 4: Fully reliable: Average goodput

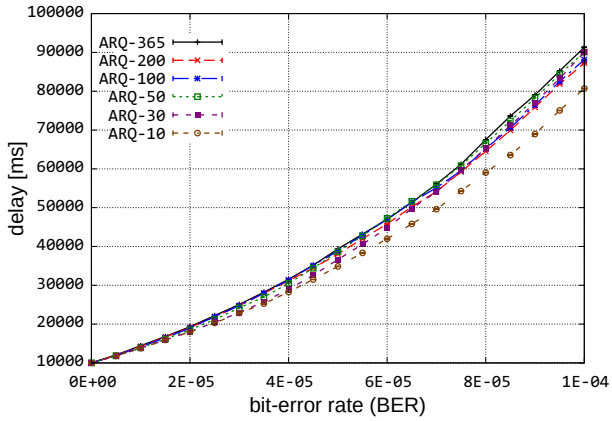

Figure 5: Fully reliable: Average delay

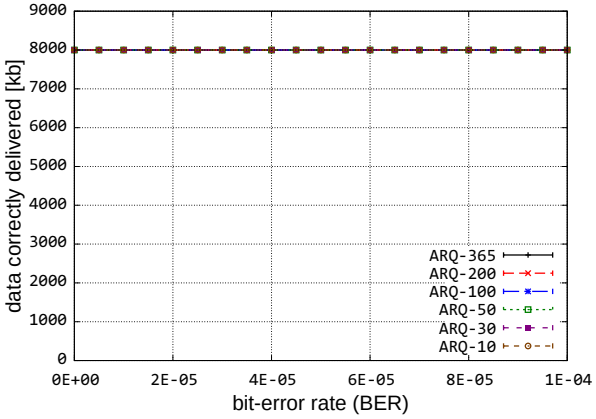

Figure 6: Fully reliable: Cumulative goodput

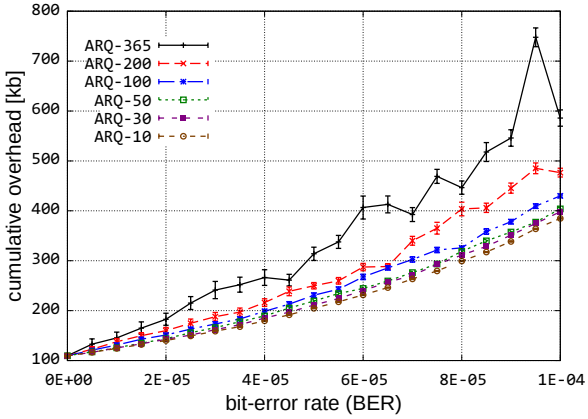

Figure 7: Fully reliable: Cumulative overhead

F. Mode Comparison over Lossy Links

In this section, we discuss the effect of increasing BER on transport layer performance metrics using different AeroTP modes, on UDP, and on TCP NewReno. The results show tradeoffs between the modes as a result of deteriorating channel conditions. Figure 8 shows that the AeroTP quasi-reliable mode is able to achieve better performance than AeroTP reliable mode, which drops off significantly as the BER increases. Both the AeroTP hyrbid-ARQ modes perform better than AeroTP reliable mode as expected. In comparison to the AeroTP protocol, TCP backs off significantly as the BER increases due to its congestion control mechanism. The UDP protocol is able to achieve better goodput compared to AeroTP reliable mode and hybrid-ARQ modes but it drops off as BER increases due to corrupted data. The AeroTP reliable mode end-to-end delay increases significantly in comparison with AeroTP hybrid-ARQ modes with a BER of 10^{-4} as shown in Figure 9. This can be attributed to the decrease in number of retransmissions due to the use of FEC codes. On the other hand, AeroTP-FEC and UDP incur no delay as they do not retransmit any data. The TCP's end-to-end delay increases nearly double than that of the AeroTP fully-reliable mode

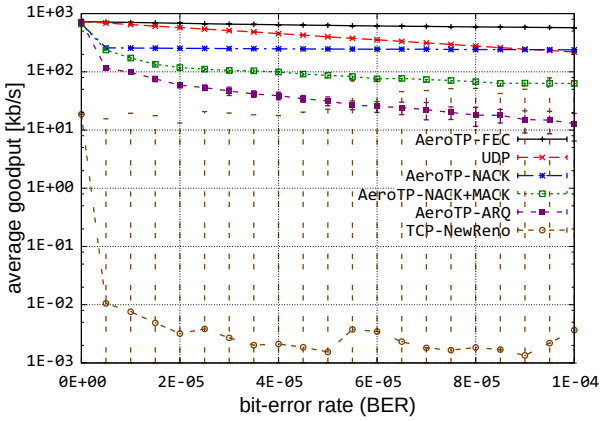

Figure 8: Mode comparison: Average goodput

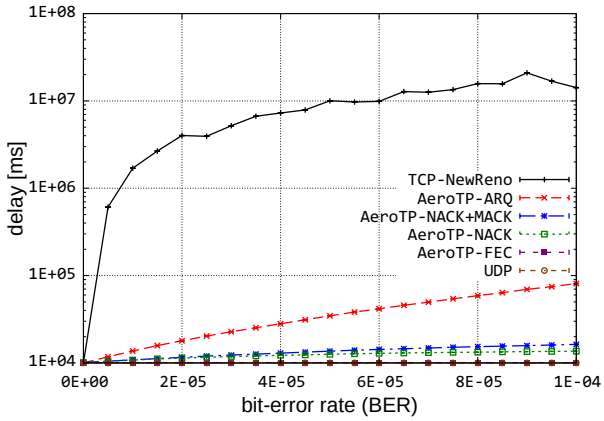

Figure 9: Mode comparison: Average delay

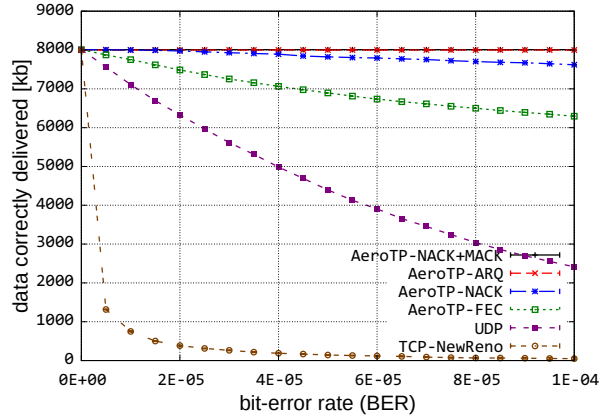

Figure 10: Mode comparison: Cumulative goodput

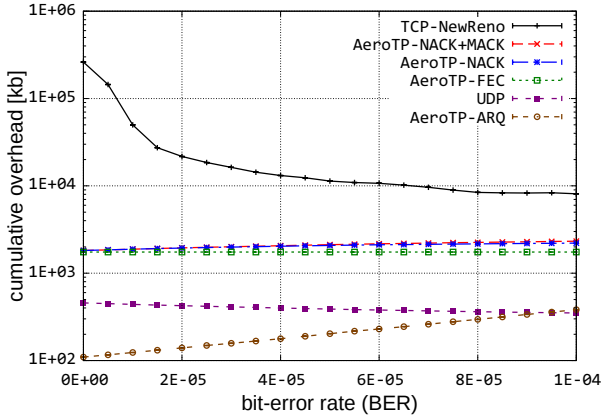

Figure 11: Mode comparison: Cumulative overhead

at BER of 10^{-4} due to higher number of retransmissions. Over the course of the simulation, the AeroTP fully reliable mode and AeroTP hybrid mode with both NACKs and MACKs enabled are able to deliver the full 1 MB of data transmitted for all BERs, but the AeroTP hyrbid mode with only NACKs enabled starts losing data for BER $> 2.5 \times 10^{-5}$, as shown in Figure 10. The plot also shows that FEC looses a higher percentage of the data due to corruption as the BER increases compared to AeroTP NACK, because the hybrid mode requests retransmissions and recovers up to a certain percentage. The UDP protocol does not employ any error detection or recovery, hence loses a percentage of data at higher BERs. Furthermore, the plot shows that the TCP is unable to deliver the full 1MB of data due to decrease in the congestion window each time a data segment is lost, after which eventually TCP connection times out. Figure 11 shows that both the AeroTP hybrid modes have larger overhead compared to a purely FEC-based quasi-reliable mode and purely-ARQ based fully-reliable mode because they have both additional FEC bits sent and data retransmitted. All the AeroTP modes incur much less overhead compared to that of TCP NewReno.

V. CONCLUSIONS AND FUTURE WORK

In this paper we presented the ns-3 simulation model of AeroTP hybrid-ARQ modes, and obtained results showing their advantages in a lossy environment. We show the tradeoffs between using AeroTP reliable mode, quasi-reliable, and the hybrid modes in terms of achieving reliability, incurring overhead, and comprimising delay and goodput. We are in the process of implementing this protocol to run in a Linux environment along with the rest of the ANTP suite [21, 22]. 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 stand 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] J. P. Rohrer, A. Jabbar, E. K. Çetinkaya, and J. P. Sterbenz, "Airborne telemetry networks: Challenges and solutions in the ANTP suite," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, (San Jose, CA), pp. 74–79, November 2010.

[2] 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.

[3] 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*, vol. 47, pp. 2742–2765, October 2011.

[4] E. K. Çetinkaya and J. P. G. Sterbenz, "Aeronautical gateways: Supporting TCP/IP-based devices and applications over modern telemetry networks," in *Proceedings of the International Telemetering Conference*, (Las Vegas, NV), October 2009.

[5] 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.

[6] K. S. Pathapati, J. P. Rohrer, and J. P. G. Sterbenz, "Edge-to-edge ARQ: Transport-layer reliability for airborne telemetry networks," in *Proceedings of the International Telemetering Conference (ITC)*, (San Diego, CA), October 2010.

[7] K. S. Pathapati, A. Nguyen, J. P. Rohrer, and J. P. Sterbenz, "Performance analysis of the AeroTP transport protocol for highly-dynamic airborne telemetry networks," in *Proceedings of the International Telemetering Conference (ITC)*, (Las Vegas, NV), October 2011.

[8] R. W. Watson and S. A. Mamrak, "Gaining efficiency in transport services by appropriate design and implementation choices," *ACM Transactions on Computer Systems*, vol. 5, pp. 97–120, May 1987.

[9] R. W. Watson, "The Delta-T transport protocol: Features and experience," *Local Computer Networks*, 1989.

[10] J. Postel, "Transmission Control Protocol." RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168.

[11] J. Postel, "User Datagram Protocol." RFC 768 (Standard), Aug. 1980.

[12] 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.

[13] 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.

[14] S. Lin, D. Costello, and M. Miller, "Automatic-repeat-request error-control schemes," *IEEE Communications Magazine*, vol. 22, pp. 5–17, December 1984.

[15] M. Rice and S. Wicker, "Adaptive error control based on type-I hybrid-ARQ protocols," in *Telecommunications Symposium, 1990. ITS '90 Symposium Record., SBT/IEEE International*, pp. 176 –180, sep 1990.

[16] H. Liu, H. Ma, M. El Zarki, and S. Gupta, "Error control schemes for networks: an overview," *Mobile Networks and Applications*, vol. 2, pp. 167–182, Oct. 1997.

[17] S. Lin and P. Yu, "A hybrid ARQ scheme with parity retransmission for error control of satellite channels," *IEEE Transactions on Communications*, vol. 30, pp. 1701–1719, July 1982.

[18] E. Y. Rocher and R. L. Pickholtz, "An analysis of the effectiveness of hybrid transmission schemes," *IBM Journal of Research and Development*, vol. 14, pp. 426–433, July 1970.

[19] D. Mandelbaum, "An adaptive-feedback coding scheme using incremental redundancy," *IEEE Transactions on Information Therory*, vol. 20, pp. 388–389, May 1974.

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

[21] M. Alenazi, 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.

[22] S. A. Gogi, D. Zhang, E. K. Çetinkaya, J. P. Rohrer, and J. P. G. Sterbenz, "Implementation of the AeroTP Transport Protocol in Python," in *Proceedings of the International Telemetering Conference (ITC)*, (San Diego, CA), October 2012. to appear.