Airborne Telemetry Networks: Challenges and Solutions in the ANTP Suite

Justin P. Rohrer, Abdul Jabbar, Egemen K. Çetinkaya, and James P.G. Sterbenz
Department of Electrical Engineering & Computer Science
Information and Telecommunication Technology Center
The University of Kansas
Lawrence, KS 66045
E-mail: {rohrej|jabbar|ekc|jpgs}@ittc.ku.edu

***Abstract*—Airborne telemetry networks are an example of a wireless network domain with much higher mobility than either the Internet or MANET protocols were designed to cope with. The very high speeds and distances covered, combined with limited radio transmission power contribute to making fully-connected operation impossible in this environment. We present challenges specific to highly-dynamic airborne networks in conjunction with the mechanisms used to ameliorate them in the Airborne Network Telemetry Protocol (ANTP) suite. At the transport layer we use multiple reliability modes, depending on the traffic class. For routing we leverage geolocation information as well as using the store-and-haul mechanism. At the border of the airborne network, we translate these protocols into the standard TCP/IP network stack using the AeroGW gateway architecture.**

I. Introduction and Motivation

Highly dynamic airborne networks pose unique challenges to end-to-end data transmission. This has become painfully evident in designing protocols for the airborne test and evaluation environment. In previous work, the environmental challenges specific to the aeronautical telemetry environment have been clearly identified [1], along with the inadequacy of current protocols to overcome them [2]. Subsequently we proposed an architecture (ANTP) to address the challenges presented [3]. In this paper we begin to fill in the details of this architecture with protocol features intended to address specific concerns. These protocols are designed to function in the context of the Integrated Network Enhanced Telemetry (iNET) program for Major Range and Test Facility Bases (MRTFB) across United States[4].

The rest of the paper is organized as follows: section II presents the opportunistic connection setup used by AeroTP. This is followed by a discussion of the AeroRP routing algorithm in section III. In section IV we describe that gateway mechanisms used to splice the Aero protocols with the standard Internet protocol stack.

II. AeroTP Connection Setup

AeroTP is a domain-specific transport protocol originally presented in [5] and designed to meet the needs of the telemetry network system (TmNS) while being *TCP-friendly*[1] to allow efficient splicing with conventional TCP at the AeroGW gateways [7] in the ground station (GS) and on the test article (TA). AeroTP performs end-to-end data transfer between the edges of the telemetry network and splices to TCP connections or UDP flows at the AeroGWs. Transport-layer functions that must be performed by AeroTP include connection setup and management, transmission control, and error control. In this section we will focus our discussion on the connection setup.

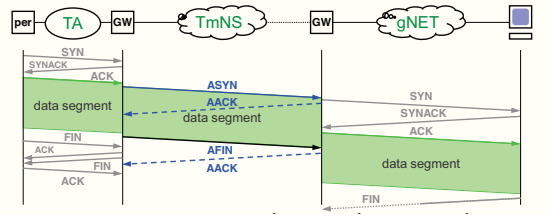


Fig. 1. AeroTP connection setup

The standard protocol used in the Internet for reliable data transfer is TCP [8], which requires a three-way handshake for connection setup. The overhead incurred by doing so is three round-trip-times (RTTs) at a minimum. If loss or corruption occurs (which it often does in wireless environments) the process must wait to timeout and try again. In an environment without stable end-to-end paths, performing this handshake may be impossible, postponing the transfer of data indefinitely. For this reason AeroTP is designed to establish a connection when the first data packet in a flow is received, allowing opportunistic connection establishment (Figure 1) in which data can begin to flow with the ASYN (AeroSYN) setup message. 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.

Preliminary simulations comparing the time required to establish a standard TCP connection with an AeroTP connection were presented in [9], and showed a significant reduction in the time required for connection establishment even with a small probability of packet loss. Within either the TA network or the GS network, systems are strongly connected, thus allowing for the use of the standard TCP connection setup algorithm. The translation between AeroTP and TCP is handled by the

[1]Note that we use the term "TCP-friendly" in a more general sense than the established term "TCP-friendly rate control" (TFRC) [6]

Aero Gateway (AeroGW) and this process is discussed in section IV-B.

III. AeroRP: Location-aware Highly Adaptive Routing Algorithm

The small contact duration among TAs results in frequent routing changes and is indicative of the need for an intelligent multihop routing protocol, supporting reliable communication over the highly dynamic physical topology. As discussed in [10], existing routing mechanisms generate significant overhead and do not converge quickly (if ever) in the presence of frequent topology changes and hence are not suitable for telemetry networks. The AeroRP routing protocol is specifically designed to address the issues related to highly mobile aeronautical environments. We utilize a number of mechanisms that have been researched independently for use in environments with characteristics similar to those of aeronautical telemetry:

1) **Proactive behavior:** AeroRP is a fundamentally proactive routing protocol, but with limited updates, and therefore low protocol overheard.
2) **Exploits cross-layer data:** AeroRP is designed to exploit the explicit cross-layering support provided by AeroNP and the geographic node location and trajectory information available at nodes.
3) **Per-hop behavior:** Unlike existing protocols, AeroRP forwards data per-hop based on partial local information and routes thereby avoiding the necessity for global convergence, making it especially suitable for highly-dynamic environments.
4) **Multi-modal:** Telemetry applications present a high level of variation in their operational parameters. For example, based on the security requirements of the test application, the geolocation of the nodes may or may not be available. In order to support these dynamics in operation, policies, and constraints, AeroRP provides multiple modes of operation.

AeroRP is based on the ANTP [11] system architecture that uses a cross-layered network protocol designed for this environment: AeroNP. The AeroNP protocol header includes fields for source and destination locations. It also has two fields to specify the quality of service of packets in the network: data *type* (e.g. command and control, telemetry) and *priority* with in a given type.

A. Protocol Operation

The basic operation of AeroRP consists of two phases. In the first phase, each node learns and makes a list of available neighbors at any given point in time. It utilizes a number of different mechanisms to facilitate neighbor discovery, discussed later in this section. The second phase of the algorithm is to find the appropriate next hop to forward the data packets. In order to forward the packets toward a specific destination, additional information such as location data or route updates is required. For each of these two phases the protocol defines a number of different mechanisms. The particular choice of mechanism to be used is dependent upon the mode of operation. The protocol does not specify a predefined set of discrete operational modes; the total number of supported modes is merely the combination of all the different mechanisms available. We now consider each of the two phases in more detail:

***1. Neighbor Discovery*:** The first objective of a node in the telemetry network is to determine its neighboring nodes. In order to achieve this, we use several different mechanisms with the objective to minimize overhead and increase adaptability. One or more of the following mechanisms may be used to populate the forwarding table depending upon the operational constraints.

- ***Active snooping*** is the primary mechanism used by the node to locate and identify its neighbors. In a wireless TDMA network, a node that is not transmitting listens to all transmissions on the wireless channel. AeroRP adds the transmitting MAC address of each overheard packet to its neighbor table. The protocol assumes cooperative nodes and symmetric transmission ranges. This implies that if a node can hear transmissions from another node, it can also communicate with that node. Stale entries are removed from the neighbor table if no transmissions from a node are heard for a predetermined time interval related to the anticipated contact duration.
- ***Hello beacons*** are used by idle nodes to advertise their presence. When neighboring nodes hear a hello beacon, they update their neighbor table appropriately. The frequency of the hello beacon is related to the minimum contact duration. For example, if the contact duration is 10 sec, the hello beacon is transmitted every second.
- ***Ground station updates*** may used to augment or replace *active snooping* in some of the telemetry scenarios, in which the ground station has a partial or even complete mission plan. The ground station sends periodic updates containing the location and trajectory vectors predicted by the mission plan to all nodes.

Security requirements may impose certain restrictions in the Aeronautical networks. In certain cases where node location or trajectory is considered sensitive, individual nodes may not include this information in the header of data packets or hello updates. In this case, the ground station may send location updates of all nodes on an encrypted channel. Finally, in the most secure mode, no geographic node information is available and the routes have to built using traditional MANET methods such as explicit routing updates and exchange of node contacts between the neighbors.

Given the dynamic nature of the aeronautical network, neighbor discovery not only consists of finding nodes within transmission range, but also determining the duration for which a discovered node will remain within range. Depending upon operational constraints, this information is obtained via different mechanisms: location and trajectory information is included in the network protocol (AeroNP) header [11], or in updates sent by the ground station.

***2. Data Forwarding*:** After neighbor discovery, the second

phase of AeroRP is for individual nodes to determine the next hop for a particular transmission. Recall that, unlike conventional protocols, AeroRP performs hop-by-hop forwarding based on partial paths without the full knowledge of the end-to-end paths. Each node forwards packets such that they end up geographically closer to the destination, which will frequently be a GS in the telemetry environment.

When any given node needs to transmit telemetry data, and assuming that one or more neighbors are discovered, the data packets are forwarded to the node that is nearest to the destination as calculated from its current coordinates and trajectory. The destination location is obtained in a manner similar to that of discovering neighbors. Furthermore, in a majority of test cases, the destination is the stationary ground station whose coordinates are known to all TAs. The actual algorithm for finding the best node to forward (or handover) the data packet is given in Section III-B

In order to avoid congestion at any given node, AeroRP utilizes the *congestion indicator* [12], [13] field of the AeroNP header. Each node uses the CI field to indicate its own congestion level. All packet transmissions from a node carry the CI field along with the type and priority of the data. All the neighboring nodes are thus made aware of the congestion at a given node for a given priority of the traffic and refrain from forwarding equal or lower priority traffic to the congested node.

B. Routing Algorithm

Let the position of i^{th} airborne node, n_i be represented by the vector $\mathbf{P}_i = (x_i, y_i, z_i)$ and the trajectory is defined by the vector $\mathbf{T}_i = (s_i, \theta_i, \phi_i)$, where x, y, and z are the absolute node coordinates, $\mathbf{T}$ is the spherical direction vector (seed, inclination, and azimuth). Since the network is highly dynamic both the position and trajectory of nodes is time dependent. For a given source–destination pair, at a given time t, let the source node n_s has the position be $\mathbf{P}_s^t = (x_s^t, y_s^t, z_s^t)$ and the trajectory be $\mathbf{T}_s^t = (s_s^t, \theta_s^t, \phi_s^t)$. Similarly the destination node n_d is represented by $\mathbf{P}_d^t = (x_d^t, y_d^t, z_d^t)$ and $\mathbf{T}_d^t = (s_d^t, \theta_d^t, \phi_d^t)$. If the destination happens to be a stationary ground station, then $\mathbf{P}_d^t = \mathbf{P}_d, \forall t$. Finally, let the congestion status of the node be given by the vector $\mathbf{C}_i = \{\text{CI}, \text{priority}\}$, where *CI* and *priority* are the congestion indicator and priority fields that are extracted from the AeroNP header.

Step 1: Each node maintains two tables: a *neighbor table* that stores the information about the nodes that are currently in the transmission range, and *destination data table* that stores the information all destinations, which may or may not be in the currently in the transmission range. Initially, the number of neighbors represented by the neighbor list N is zero, i.e. $N = \emptyset$.

Step 2: When the node receives any transmission, it updates the neighbor and destination data tables. If the captured packet is an overheard transmission or hello advertisement from node n_i, the node i is assumed to be in the transmission range of the current node. Hence the neighbor list is updated as $N = N \cup \{n_i\}$. Furthermore, the macID, position, trajectory, and congestion status of the node are derived from its header and stored in the neighbor table as the tuple $\{\text{macID}_i, \mathbf{P}_i^t, \mathbf{T}_i^t, \mathbf{C}_i^t\}$. If the received transmission is a *ground station update*, each entry in the update is stored in the destination data table as the tuple $\{\text{time}, \text{macID}_i, \mathbf{P}_i^t, \mathbf{T}_i^t, \mathbf{C}_i^t\}$. Since the GS update may contain information on node positions in future, the entries in the destination data table are time stamped. Lastly, when a ground station update is received, the location and trajectory fields of neighbor table entries are updated with the latest values.

Step 3: At the completion of step 2, assume that a given node n_0 has k discovered neighbors. Hence $N_0 = \{n_0, n_1, \ldots, n_i, \ldots, n_{k-1}\}$. Note that each node treats itself as the first neighbor. Assume that this node n_0 wants to send a data packet to the ground station n_d with position $\mathbf{P}_d$. Assume that the transmission range of all nodes is R. Next, we calculate the *time to intercept*, TTI for all neighbors. The TTI_i represents the time it will take for node n_i to get reach within the transmission range of the destination if it continues on its current trajectory. TTI is calculated as:

$$\text{TTI}_i = \frac{|\mathbf{P}_d^t - \mathbf{P}_i^t| - R}{s_d} \tag{1}$$

where $|\mathbf{P}_d^t - \mathbf{P}_i^t|$ gives the euclidian distance between the current location of node n_i and the destination node n_d and s_d is the component of the actual speed r of node n_i in the direction of the destination.

Step 4: Finally, the data is forwarded to the j^{th} node, n_j such that:

$$\text{TTI}_j = \min\{\text{TTI}_i\} \quad \forall i : n_i \in N_0 \tag{2}$$

The process is repeated at every node, until the data reaches the destination.

IV. AeroGW: Interface to the TCP/IP Protocol Suite

This section describes the AeroGW functionality. The source and destination of telemetry data are expected to be TCP/IP-based systems; however the TmNS (telemetry network system) will employ new domain-specific protocols suitable for the dynamic airborne environment. To overcome this challenge without requiring a total redesign of all telemetry sensors, peripherals, applications, and workstations, we introduce the Aero Gateway (AeroGW). The protocol architecture is shown in Figure 2, showing the gateway functionality at the TA and GS, with optional intermediate relay nodes (RNs).

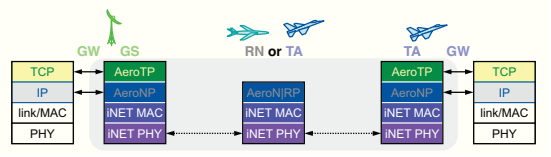


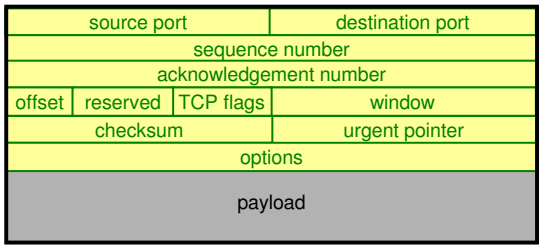
Fig. 2. System Architecture

Gateways can be thought of as an interface that provides translation between two entities, can support different access

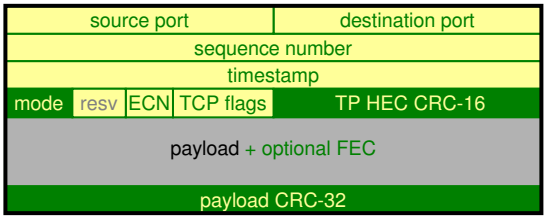
technologies or different protocols, and generally have sufficient processing resources. The gateway concept is well established [14] as a mechanism for bridging between disparate network environments. In this case its operation is similar to TCP-Splice [15], however instead of splicing TCP with TCP, it will splice TCP (and UDP/RTP) to AeroTP and IP to AeroNP. This translation functionality resides in the AeroGW, which is incorporated into the ground network (gNET) interface element on the ground station and the vehicle network (vNET) element on each TA.

A. TCP – AeroTP Splicing

AeroTP is the transport layer proposed [5], [9] to overcome the challenges of the highly dynamic wireless, mobile aeronautical environment. It provides opportunistic connection setup and management, transmission control, and flexible error control functions. AeroTP supports different modes of operation: reliable, nearly-reliable, quasi-reliable, best-effort connections, and best-effort datagrams. Each of the operational modes is designed for the needs of specific missions. AeroTP is TCP friendly in that it allows efficient splicing with TCP at the gateways. TCP is well documented in numerous RFCs [8], [16]–[18]. Excluding the 4 byte long optional fields, TCP adds a 20 byte header per packet as shown in Figure 3(a). This is a substantial overhead in the case of control traffic such as ACKs. For a side by side comparison, the proposed AeroTP header format is shown in Figure 3(b). It should be noted that the light colored fields are from the TCP header, while the dark colored fields are new in the AeroTP header and trailer.



(a) TCP Segment



(b) Aero TPDU

Fig. 3. Transport Layer Protocol Data Units

The AeroTP header consists of a 16 byte header. Source and destination port fields identify the sending and receiving ports. The sequence number allows reordering of packets due to erasure coding over multiple paths or TA mobility, and is either the TCP byte-sequence number or an AeroTP TPDU number, depending on the transfer mode. The timestamp field is 32 bits wide for end-to-end TCP transparency. There are five modes of operation to match the reliability needs of a telemetry application. The Header Error Check (HEC) field is a strong Cyclic Redundancy Check (CRC) on the integrity of the header to detect bit errors in the wireless channel. This allows the packet to be correctly delivered to AeroTP at the destination when a corrupted payload can be corrected on an end-to-end basis using FEC. A CRC trailer protects the integrity of the data edge-to-edge across the telemetry network in the absence of a separate AeroNP or link layer frame CRC and enables measurement of the bit-error-rate for erasure code adaptation depending on the transfer mode.

TCP requires a three-way handshake process to establish a connection before data is transmitted. This connection setup consumes 1.5 round trip times (RTTs) and prevents the sending of any data before a stable end-to-end path exists. AeroTP uses connection management paradigms suited to the telemetry 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 AeroSYN (ASYN) setup message, which will not waste the 1.5 RTT. The opportunistic connection signaling paradigm for networked telemetry system is shown in Figure 1.

A packet must pass through two gateways on its path from source to destination. The ingress gateway will convert the TCP messages to AeroTP messages, while egress gateway will convert AeroTP messages to TCP format. It should be noted that ingress and egress gateways are not additional network elements in the TmNS environment, but rather the gateway functionality will be built in TAs and GSs. The flow diagram for the TCP to AeroTP translation that occurs at the ingress gateway is presented in Figure 4. Essentially, ingress gateway will splice the end-to-end TCP protocol. Once the TCP SYN message is received, gateway will send back a SYN ACK message. Upon receiving the SYN ACK message the source will send the TCP ACK message. The gateway will transmit the ASYN message along with the data to the TmNS after receiving the TCP ACK message. The data can piggyback the ASYN message. The ingress gateway will check the successful transmission of the data to the egress gateway via incoming AeroACK (AACK) messages. If the ASYN message is delivered to the egress gateway, data can continue to flow from source to destination. In the case of a failed delivery of the ASYN message, it should be sent again to preserve the end-to-end TCP semantics. Once the destination receives the application data, it will send a TCP FIN message to the gateway signaling termination of the connection. The egress gateway will send the corresponding AeroFIN (AFIN) message to the ingress gateway to end the connection.

The flowchart for the AeroTP to TCP translation that occurs at the egress gateway is shown in Figure 5. The egress gateway complements the splicing function by reconstructing the TCP messages. Upon receiving the ASYN message, the egress

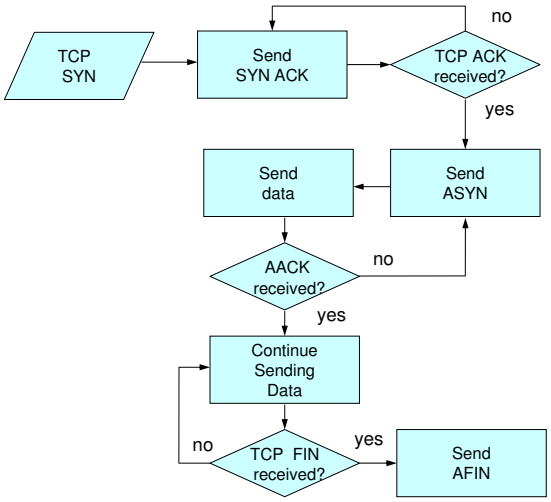


Fig. 4. TCP to AeroTP Conversion

gateway will send the TCP SYN message to the destination. Delivery of the TCP SYN message is checked with the SYN ACK message. If SYN ACK is not received, the egress gateway will retransmit the TCP SYN message. Upon receiving the SYN ACK, the egress gateway can start transmitting the data, which includes the ASYN and application or control data it received from the ingress gateway. Once the TCP FIN message is received from the destination, the egress gateway will transmit the AFIN message to the TmNS for connection termination.

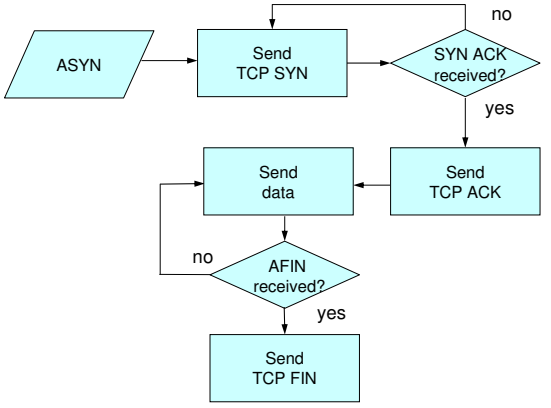
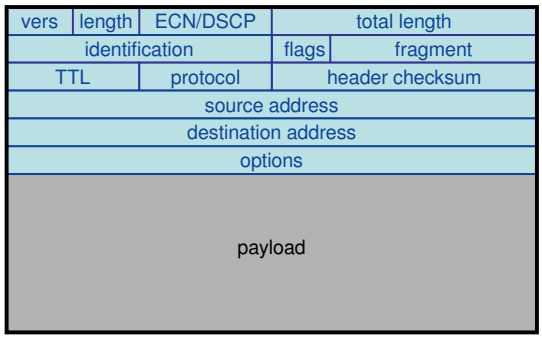


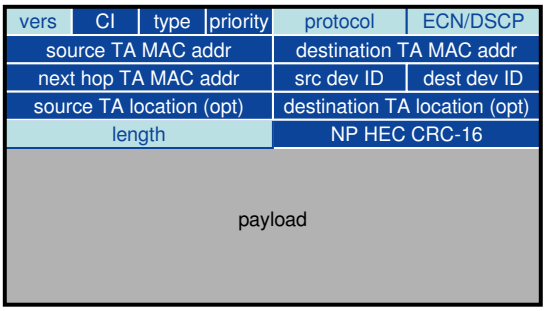
Fig. 5. AeroTP to TCP Conversion

B. IP – AeroNP Translation

The network layer provides services to the transport layer. To adapt the needs of AeroTP a new network protocol, AeroNP [11], is proposed for the telemetry environment. AeroNP is *IP-compatible* meaning that, given the IP-based end devices such as HMI applications on the ground network and the TCP/IP peripherals on the TA, it is critical for the network protocol on the telemetry subsystem to be interoperable with IP. IP is the de facto network layer protocol for the Internet and its operation is standardized and well documented in RFCs [19], [20]. Excluding the optional fields, the IP header is 20 bytes long and its format is shown in Figure 6(a). The proposed AeroNP header is 32 bits wide, shown in Figure 6(b).



(a) IP Packet



(b) Aero NPDU

Fig. 6. Network Layer Protocol Data Units

In the Aero NPDU the light colored fields come from the IP header and dark colored fields are specifically designed for the AeroNP header. The congestion indicator field is set by each node to notify the neighboring nodes of its congestion level. This field provides support of reliable communication paths in multi-hop scenarios in which the node that is congested should not be considered for path setup by the AeroRP routing algorithm. The optional source TA location and destination TA location fields designed to be used by the AeroRP. These fields contain the GPS coordinates that are used in location-aware routing [10].

Given that wireless links are bandwidth constrained and different applications have their own requirements to operate, it is essential to have a quality of service mechanism. The type and priority fields specify the QoS level of a given packet. The ECN/DSCP field is also used for QoS mechanisms, but this field is carried for end-to-end IP transparency. Version, protocol, and length are the other AeroNP fields that will passed in the gateways without any modification for IP-compatibility.

AeroNP does not carry the 32-bit source and destination IP address fields. Instead, it utilizes the iNET MAC address of existing hardware. Since iNET MAC is based on TDM, an AeroNP packet is inserted directly into a TDM slot. Some header space efficiency will be gained by eliminating the use of

network address fields in the header. However to be compatible with the existing IP based services, it is clear that an IP address to AeroNP MAC address translation is necessary at the gateways. This address conversion mechanism can be similar to Address Resolution Protocol (ARP) function. In ARP [21], a table is constructed such that there is a mapping between an IP and MAC address. The next hop TA MAC address field will be used by the AeroRP for identifying the route to the next hop.

TAs can have multiple peripherals for different purposes. Each of the peripherals will use a separate IP address. Since the AeroNP header does not have IP address fields, each peripheral can be mapped to a device id. This translation functionality is similar to Network Address Translation (NAT) [22] and resides at the gateways. While IP address to device IDs can be mapped dynamically, it is more efficient to preload the mapping table at the beginning of each mission. Finally, HEC field is used to check the integrity of the AeroNP header fields against bit errors.

AeroNP is IP compatible to support end-to-end transparency to TCP/IP based applications. The primary translation required is for IP address to MAC address, and IP address to device ID mapping. Both of the translation and mapping tables can be a preprogrammed for efficiency and simplicity.

V. Conclusions and Future Work

It has been noted in previous works that the existing Internet protocol architecture is not well suited for applications in highly-dynamic airborne tactical networks, which present unique challenges due to extreme mobility and limited bandwidth. Additionally, typical MANET routing protocols such as AODV and DSDV are not designed for topologies that are as dynamic as the ones found in aeronautical environments. In this paper, we have described several transport and routing mechanisms that address these issues within the ANTP suite. As we continue to develop these protocols, we will be examining the interactions between the multiple layers, and the cross-layer exchange of information needed to facilitate the functionality described in this paper. We intend to perform more extensive simulations with varying node densities, mobility models, etc. Eventually we will be able to simulate the entire Aero protocol suite simultaneously. With the models refined iteratively through the simulation process, we will proceed to implementing prototypes of the entire Aero suite for field testing.

Acknowledgments

We would like to acknowledge Kip Temple and the members of the iNET working group for discussions that led to this work. This research was funded in part by the International Foundation for Telemetering (IFT), and by the National Science Foundation FIND (Future Internet Design) Program under Grant No. CNS-0626918 (PoMo).

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] J. P. Rohrer, A. Jabbar, E. Perrins, and J. P. 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), November 2008.

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

[5] J. P. Rohrer, E. Perrins, and J. P. 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] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification." RFC 3448 (Proposed Standard), Jan. 2003. Obsoleted by RFC 5348.

[7] E. K. Çetinkaya and J. P. 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 26–29 2009.

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

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

[10] A. Jabbar and J. P. Sterbenz, "AeroRP: A geolocation assisted aeronautical routing protocol for highly dynamic telemetry environments," in *Proceedings of the International Telemetering Conference*, (Las Vegas, NV), October 26–29 2009.

[11] A. Jabbar, E. Perrins, and J. P. 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.

[12] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 158–181, 1990.

[13] M. Mamun-Or-Rashid, M. M. Alam, M. A. Razzaque, and C. S. Hong, "Congestion avoidance and fair event detection in wireless sensor network," *IEICE Trans Commun*, vol. E90-B, no. 12, pp. 3362–3372, 2007.

[14] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations." RFC 3135 (Informational), June 2001.

[15] D. A. Maltz and P. Bhagwat, "TCP Splice for application layer proxy performance," *Journal of High Speed Networks*, vol. 8, no. 3, pp. 225–240, 1999.

[16] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options." RFC 2018 (Proposed Standard), Oct. 1996.

[17] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control." RFC 2581 (Proposed Standard), Apr. 1999. Obsoleted by RFC 5681, updated by RFC 3390.

[18] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP." RFC 3168 (Proposed Standard), Sept. 2001.

[19] J. Postel, "Internet Protocol." RFC 791 (Standard), Sept. 1981. Updated by RFC 1349.

[20] R. Callon and H. Braun, "Guidelines for the use of Internet-IP addresses in the ISO Connectionless-Mode Network Protocol." RFC 1069, Feb. 1989.

[21] D. Plummer, "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware." RFC 826 (Standard), Nov. 1982. Updated by RFCs 5227, 5494.

[22] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)." RFC 3022 (Informational), Jan. 2001.