Aero-Sim: An NS-2 Based simulator for Aeronautical Ad Hoc Networks

Qin Luo[1], Junfeng Wang[2], Xiaoqing Wang[3] and Ke Wu[2]
[1] School of Aeronautics and Astronautics, Sichuan University
Chengdu 610064, P.R. China
[e-mail: dorothy_lq@163.com]
[2] College of Computer Science, Sichuan University
Chengdu 610064, P.R. China
[e-mail: wangjf@scu.edu.cn, wuke_1992@126.com]
[3]Beijing Institute of System Engineering
Beijing 100101, P.R. China
[e-mail: wang_xiaoqing@126.com]
*Corresponding author: Junfeng Wang

Received September 13, 2014; revised May 8, 2015; accepted June 2, 2015; published July 31, 2015

Abstract

Recently, there has been a clear trend towards the application of ad hoc networking technology in civil aviation communication systems, giving birth to a new research field, namely, aeronautical ad hoc networks (AANETs). An AANET is a special type of ad hoc wireless network with a significantly larger scale and distinct characteristics of its mobile nodes. Thus, there is an urgent need to develop a simulator to facilitate the research in these networks. In this paper, we present a network simulator, Aero-Sim, for AANETs. Aero-Sim, which is based on the freely distributed NS-2 simulator, enables detailed packet-level simulations of protocols at the MAC, link, network, transport, and application layers by composing simulations with existing modules and protocols in NS-2. Moreover, Aero-Sim supports three-dimensional network deployment. Through several case studies using realistic China domestic air traffic, we show that the proposed simulator can be used to simulate AANETs and can reproduce the real world with high fidelity.

Keywords: Aeronautical ad hoc networks; NS-2; network simulation; aeronautical communicaiton

This work was supported by the National Natural Science Foundation of China (91338107, 91438119), the Ph.D. Program Foundation of Ministry of Education of China (20130181110095), the Jiangsu Future Networks Innovation Institute Prospective Research Project on Future Networks (BY2013095-3-08).

http://dx.doi.org/10.3837/tiis.2015.07.012 **ISSN : 1976-7277**

1. Introduction

In recent years, there has been substantial growth in mobile ad hoc networks (MANETs) in land-based small-to-medium size networks with relatively strict power and resources. However, MANETs should not be limited to small devices and localized parameters, such as Bluetooth devices and IEEE 802.11 wireless local area networks [1]. They can be extended almost to a global network level where the mobile entities are large-scale systems with relatively large resources and transmission power. Therefore, some research communities and researchers have begun applying MANETs to aeronautical communications [2] [3] [4] [5], leading to the rise of a novel research field, aeronautical ad hoc networks (AANETs) [3]. An AANET is a special type of MANET, where the nodes represent aircraft in an airspace for the purpose of sharing data and Internet access. In an AANET, an aircraft is not only a transceiver, but also a router that can forward packets to other aircraft through multi-hop communications. As the future development direction of military and civil aviation communication, research on and application of AANETs is promising.

Network simulators, which can provide a controlled environment for researchers, are widely used in researching various types of networks when hardware resources are limited. And network simulation is undoubedly one of the most widely used evaluation methodologies for research on computer networks. It is widely used as the first step in evaluating the functionality of the newly proposed designs for network research as well as for developing and validating the new protocols. Thus, it is highly desirable to have a standard network simulator to facilitate research on AANETs.

To the best of our knowledge, there is no complete packet level aeronautical simulator published yet. Although there are several widely used packet-level simulators such as Network Simulator 2 (NS-2) [6] and Optimizing the Performance of the Network (OPNET) [7], these were developed for wired and/or terrestrial wireless networks, and not for aeronautical networks. They cannot be used for simulating aeronautical networks for the following reasons. First, AANETs are implemented in a three-dimensional (3D) environment, with nodes moving over large distances, whereas these simulators typically support only two-dimensional deployment of MANETs. For example, in the NS-2 network simulator, Carnegie Mellon University (CMU)'s Monarch research groups [8] have contributed support for MANETs [9]. The module essentially consists of a *MobileNode* at the core, which is a basic *Node* object with added functionalities of a wireless and mobile node, such as ability to move within a given topology, receive and transmit signals to and from a wireless channel, and so on. A *MobileNode* is designed to move in a 3D topology; unfortunately, the third dimension is not used. The satellite module [10] in NS-2 can support 3D deployment of network nodes; however, the module does not define aeronautical nodes and their attributes. In addition, no protocols suitable for AANETs have been designed and implemented in this module. Second, the high speeds of the aircraft cause network topologies to change rapidly, which can have a significant impact on network connectivity, and the media access control (MAC) and routing protocols. The corresponding protocols in these traditional simulators are not well suited to AANETs and new protocols need to be incorporated. These characteristics make existing network simulators unsuitable for use with AANETs.

In this work, we developed a simulator, called Aero-Sim, for AANETs. The simulator is based on NS-2, the most popular network simulator in both academia and industry and which has become the de facto standard for simulation of packet-switched networks [11] [12]. Since

NS-2 is an open-source event-driven simulation tool, anyone can make changes to the existing code or incorporate new simulation modules, extending the built-in NS-2 code. Meanwhile, researchers can make full use of existing protocols and utilities in NS-2 for their own development. Thus, we chose NS-2 as the development platform for simulating AANETs.

Based on NS-2, Aero-Sim supports a 3D aeronautical environment and can faithfully simulate the attributes of aeronautical nodes. Moreover, by composing simulations with existing modules and protocols(such as transmission control protocol (TCP) or user datagram protocol (UDP)) in NS-2, Aero-Sim enables packet-level simulations of protocols at the MAC, link, network, transport, and application levels of the TCP/IP reference model. Furthermore, it can easily be integrated with other aspects of NS-2. Thus, Aero-Sim can simulate AANET communication very well.

In this paper, we report on our extension of NS-2, namely, the AANET module. Our main contributions are the following. First, we propose an integrated module based on NS-2 that ensures compatibility with existing modules and allows easy integration of new protocols as the simulator for AANETs. Second, based on an analysis of the characteristics of AANETs, we clearly define the structure of an aeronautical node. Finally, we systematically implement protocols in the MAC and routing layers that are suitable for aeronautical communications.

The remainder of this paper is organized as follows. Section 3 details the design goals and overall structure of the simulator. The design and implementation of the simulator are presented in detail in Section 4. Through implementation of various case studies, Section 5 illustrates some of the capabilities of our simulator. Finally, we present our conclusions and suggestions for future work in Section 6.

2. Overview of Aero-Sim

In this section, we first describe the design goals of the simulator. Thereafter, we give an overview of the Aero-Sim design, and discuss its motivation, overall structure, and the relationship to the NS-2 simulator.

2.1 Design Goals

To ensure that the proposed simulator can be easily configured, used, and extended for a wide range of applications, the following design goals were identified:

- *Modular design:* Design of the simulator must be highly modular to ensure that different parts of the simulator have minimal interdependencies. Meanwhile, the simulator must be independent of other aspects of the NS-2 simulator and not be affected by any changes in the NS-2 simulator. Likewise, any changes to Aero-Sim should be confined to its own code and should not have any impact on other packages in NS-2.
- *Framework extensibility:* As techniques advance, new models and protocols may be created for integration with aeronautical networks. A highly extensible framework enables design, integration, and use of new models with minimal modifications to the basic framework structure.
- *Ease of use:* The models included in the simulator must be easy to configure in NS-2 simulations. The parameters in the simulator can be configured with different values to reproduce different scenarios.

2.2 Basic Structure of the Simulator

As discussed earlier, in NS-2, the CMU wireless module can support simulation of terrestrial wireless ad hoc networks. Nevertheless, such a simulation package cannot easily be applied to aeronautical networks. Thus, we introduced several new components into NS-2 to model aeronautical networks more accurately.

To meet the requirement of modular design, instead of adding an existing wireless networking module, we developed a new NS-2 module, called Aero-Sim, which exists in parallel with the CMU wireless module and other NS-2 modules. **Fig. 1** illustrates the relationship between Aero-Sim, the CMU wireless package, and other NS-2 modules. In this way, Aero-Sim can be compatible with other aspects of the NS-2 simulator, but can also evolve independently, thereby making the simulator highly modular.

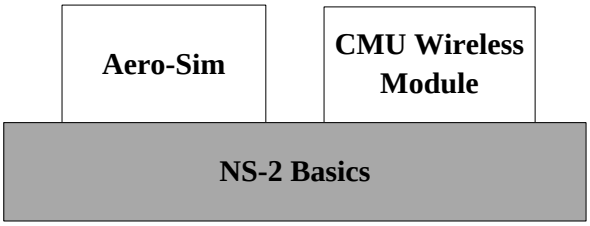

Fig. 1. Relationship between Aero-Sim and other modules in NS-2

Aero-Sim follows the object-oriented design style of NS-2, and all network entities are implemented as classes in C++. Object tool command language (OTcl) scripts are used to facilitate tuning of parameters of the protocols and algorithms implemented in C++. Such a design makes the integration of new protocols with Aero-Sim as well as the adjustment of protocol parameters easy. In other words, the basic structure of Aero-Sim satisfies the design goals well. As we can see in **Fig. 2**, the source C++ code is located in the *~ns/Aero-Sim* directory and the OTcl code is located in the *~ns/tcl/lib* directory. The folder aero-test includes all the OTcl script examples to validate Aero-Sim.

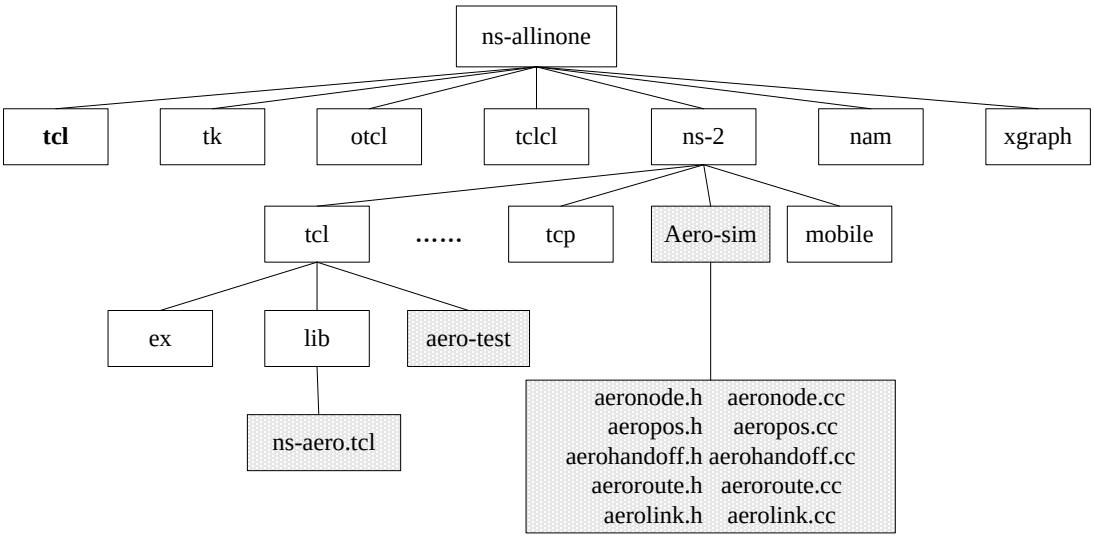

Fig. 2. NS-2 directory structure including the Aero-Sim module

3. Design and Implementation

In this section, based on the design goals, we present the design and implementation of Aero-Sim in detail, including the aeronautical node classes, node position classes, node handoff classes, MAC layer classes, and routing layer classes.

3.1 Aeronautical Nodes

There are three types of aeronautical nodes in the Aero-Sim module, with the most common type representing aircraft (A/C). Besides A/C, two other node types are available: airports and satellites. Although satellites are considered part of the network, their use is limited to areas where other means of communication are unavailable.

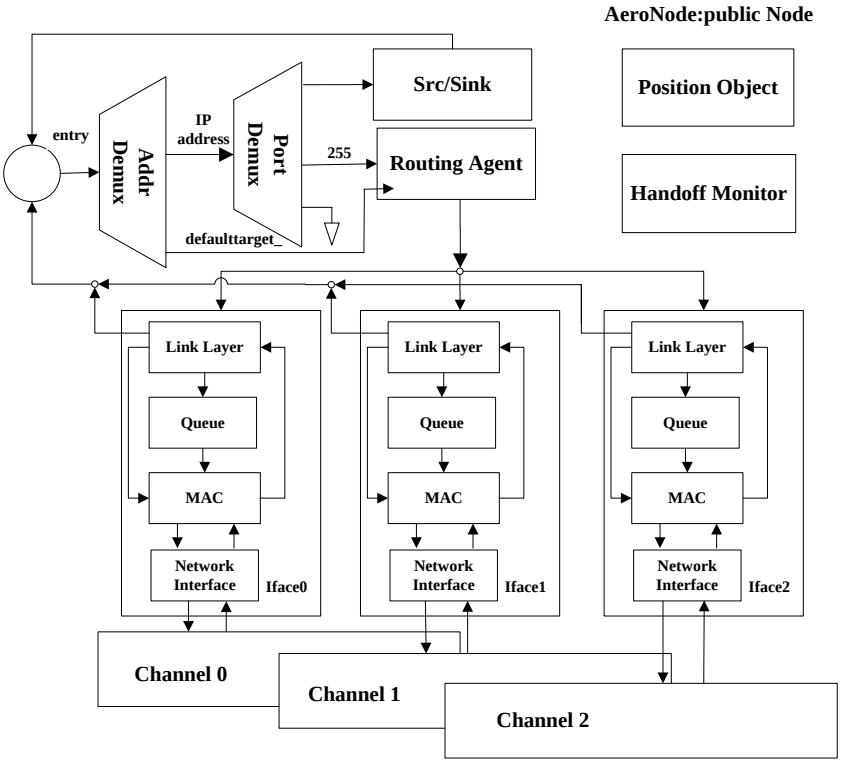

Fig. 3. The structure of class AeroNode

The detailed implementation of class *AeroNode* resides in the *aeronode* files. **Fig. 3** depicts a schematic of the main components of an *AeroNode*. As can be seen, this architecture of *AeroNode* can support the aeronautical node with multiple interfaces and each node would have as many copies of the chain of protocol stack entities as many interfaces it has. The *AeroNode* object is an abstraction of all three aeronautical nodes types. Instances of each of the nodes are actually implemented using the same node object, but with different position, handoff manager, and link objects attached.

3.1.1 Position Object

To keep track of the aeronautical node's location, we add an *AeroPosition* object to an *AeroNode* node. Specifically, there is a single class of aeronautical node Class *Node/AeroNode*, to which one of four types of *AeroPosition* objects may be attached. Each

AeroNode and *AeroPosition* object is a split OTcl/C++ object, but with most of the code written in C++. All the classes related to the *AeroPosition* object are implemented in the *aeropos* files, while all position objects are derived from the same abstract base class *AeroPosition*. Four child classes are derived from the *AeroPosition* base class: *AircraftAeroPosition*, *AirportAeroPosition*, *GeoAeroPosition* and *PolarAeroPosition*. Public interfaces that are common to all position objects are specified in *AeroPosition.* For example, if one wished to know the location of an aircraft at any given moment, the common "coord" interface, specified in class *AeroPosition*, could be invoked without any knowledge of the detailed implementation. In this way, a new position object could easily be imported into Aero-Sim by implementing the interface specified in *AeroPosition*.

Below we give a brief overview of the four *AeroPosition* objects in Aero-Sim.

- *AeroPosition/Geo and AeroPosition/Polar:* These position objects track the location of the Geo and Polar satellites. The implementation method is similar to that included in the satellite module in NS-2.
- *AeroPosition/Airport:* The location of an airport is specified by its latitude and longitude. As the simulation time advances, the airports move along with the Earth's surface. We can attach an AeroPosition/Airport object to an airport node in an OTcl script as follows:

  $airport0 set-position $lat $lon;
- *AeroPosition/Aircraft:* The location of an aircraft is specified by the time it takes off and lands, flight altitude and flight route. Flight trajectories are idealized by interpolating between two points on the flight route using great circle arcs, corresponding to the shortest distance between two points on the surface of a sphere. We can attach an AeroPosition/Aircraft object to an aircraft node in an OTcl script as follows:

  $aircraft0 set-position $opt(time19_30) $opt(time22_20) $opt(alt_aircraft) ${airlineZUUU-ZBAA};

  where ZUUU and ZBAA denote the International Civil Aviation Organization (ICAO) codes for the Chengdu and Beijing airports, respectively.

Because the AANET is implemented in a 3D environment, we use a spherical coordinate system to compute the location of a node. The main parameters of the coordinate system are defined in the *s_coordinate* data structure shown below.

```
struct s_coordinate {
    double r;       // km
    double theta;   // radians
    double phi;     // radians
};
```

This coordinate system is centered at the Earth's center, with the z-axis coinciding with the Earth's rotational axis. However, this can easily be converted to a Cartesian coordinate system using a node's latitude and longitude. Therefore, an aeronautical node's location at any given moment can be calculated. For example, assume that an aircraft is flying from Chengdu to Beijing. For simplicity of computation, we selected three navigation points on this flight route: ONEBA, OKVUM and ISGOD. The latitude and longitude coordinates of the airports and navigation points can be set in the OTcl script configuration, and the aircraft's instant position can be described as *(A. r, A. theta, A. phi)* [see **Fig. 4**].

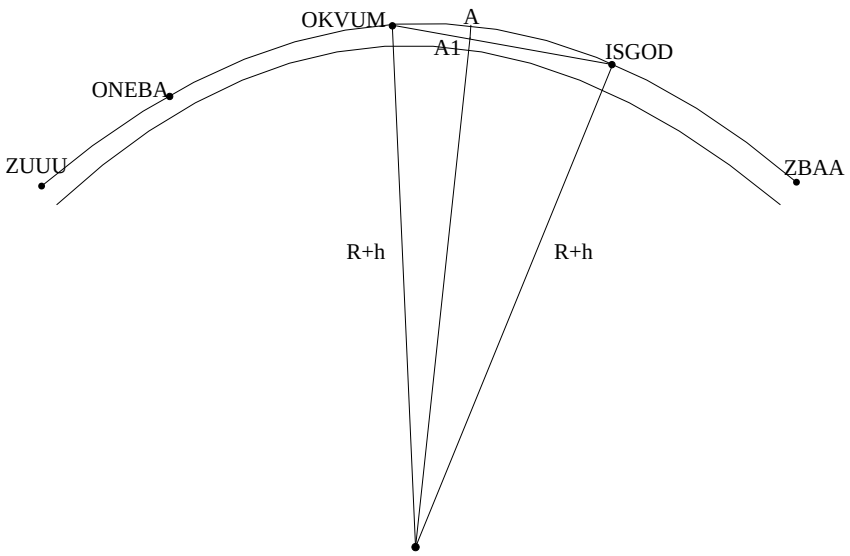

Fig. 4. Theoretical model to calculate the position of an aircraft

$$A.r = R+h;$$
$$A.theta = A1.theta;$$
$$A.phi = A1.phi; \quad (1)$$

Here, R is the earth's radius, h is the average altitude of the aircraft, *A1.theta* and *A1.phi* can be obtained through the position parameters of the navigation points and airports, the speed of the aircraft, the current time, and the corresponding coordinate transformation. Finally, the impact of the Earth's rotation on a node's location is also considered. The values of *A.r* and *A.theta* are unchanged, but the value of *A.phi* is changed as follows when considering the Earth's rotation:

$$A.phi = fmod((A.phi + earth\_w * (NOW - starttime)), 2*PI) \quad (2)$$

where earth_w is the angular velocity of the Earth's rotation, NOW is the current time, and starttime is the starting time of the flight.

3.1.2 Handoff Monitor

Unlike terrestrial wireless ad hoc networks, in an aeronautical environment, aircraft move at extremely high speeds, with the topology changing quickly. Since the links between aeronautical nodes need to be dynamically detached and attached to different nodes, we introduced AeroHandoff agents to manage link handoffs.

In the Aero-Sim module, we establish certain criteria for performing handoffs, and allow nodes to independently monitor a handoff. All the C++ codes for dealing with handoffs resides in the *aerohandoff* files. We defined three classes to handle handoff of aeronautical nodes: *AircraftLinkHandoffMgr*, *AirportLinkHandoffMgr* and *SatLinkHandoffMgr*, which are derived from the same abstract base class *AeroLinkHandoffMgr*. The child classes implement the virtual function *handoff()* in class *AeroLinkHandoffMgr*, based on the different handoff policies for different types of aeronautical nodes. Since the satellite handoff policies have been implemented in the satellite module, here we discuss the handoffs of the aircraft. There are

three types of aircraft links that can be handed off: AGLs for links from an aircraft to an airport, AALs for links between aircraft and ASLs for links from an aircraft to a polar satellite.

```
int AircraftLinkHandoffMgr::handoff()
{
  if( linktype == LINK_AGL)
  {….}
  if(linktype == LINK_AAL)
  {….}
  if(linktype == LINK_ASL_POLAR)
  {….}
}
```

In this function, different handoff policies have been implemented based on the type of the link. An aircraft connected to an airport runs a timer that, upon expiry, causes the *AeroHandoff* manager to check whether the distance between the aircraft and the airport is greater than the maximum communication distance. If so, the handoff manager disconnects the aircraft from that airport's interface, and searches for another possible airport. If it finds a suitable airport, it connects its network interfaces to that airport's uplink and downlink channels, and restarts the handoff timer. If it does not find a suitable airport, it restarts the timer and tries again later. These policies are adapted for AAL links as well. ASL links execute a handoff in the same way as GSL links from a terminal to a polar satellite in the satellite module.

As is widely known, in aeronautical communication, an aeronautical node is assumed to possess an abundance of power. The theoretical maximum transmission range between two nodes is limited by the horizon for typical line-of-sight communication in the very high frequency (VHF) band. Assuming a flight altitude of 10 km, according to the curvature of the earth, air–air and air–ground links can extend to approx. 600 km and 400 km, respectively, which is on par with current and future aeronautical VHF specifications.

3.2 Physical Layer and MAC Layer

All classes related to the physical and MAC layers are included in the *aerolink* files. Since we were more interested in research on protocols above the physical layer, we chose to abstract out the precise details of physical motion and channel modeling in Aero-Sim. Similar to the code in the wireless module, class *AeroPhy* merely passes the information up and down the stack.

```
void AeroPhy::sendDown(Packet *p)
{
 /* pass the packet to the channel */
   if(channel_)
     channel_->recv(p, this)
 /* drop the packet if there is no channel */
  else
    packet::free(p);
}
int AeroPhy::sendUp(Packet *p)
{
```

```
/*no radio propagation model be attached here, since the details of physical channel
modeling have been abstracted out */
Return TRUE;
}
```

Ho Dac Tu et al. [13] [14] proposed the widely used carrier sense multiple access (CSMA) [15] MAC protocol, which cannot be exploited effectively in AANETs considering the wide area communication network with a radius of 600 km. Currently, many AANET projects use the time division multiple access (TDMA) protocol in the MAC layer. For these reasons, the MAC protocol used in Aero-Sim is a simple reservation TDMA scheme, with the TDMA frame format illustrated in **Fig. 5**:

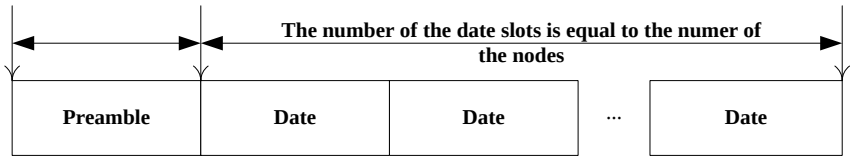

Fig. 5. The structure of a TDMA frame

.In the preamble time slot, every node uses a special sub-time slot to broadcast the destination MAC address of the packet to be sent. Other nodes listen to the preamble and record the time slot from which they receive data from. Meanwhile, every node has a data time slot to send data.

The data structure of the preamble time slot is designed and initialized as follows:

```
static int * preamble_tdma;
/* max_slot_num is the number of the nodes */
preamble_tdma = new int[max_slot_num];
/*slot_num stands for the number of the time slot, dst stands for the mac address of the
destination node */
preamble_tdma[slot_num] = dst;
```

The main task during the preamble time slot is to set the value of array *preamble_tdma[]*, which denotes the destination MAC address of the data that will be sent during time slots zero to *max_slot_num-1*. When the preamble time slot finishes, nodes send data in their own data time slots, and decide whether to receive data in other time slots based on the information in the preamble. The following functions are the implementation of the send and receive processes:

```
/* send the packet */
Void TdmaMac::send () {
…
/*get the transmit time of the packet */
int packetsize_ = HDR_CMN(Txpkt_)->size() + LINK_HDRSIZE;
assert(bandwidth_!=0);
txt = txtime(packetsize_);
HDR_MAC(p)->txtime() = txt;
…
```

```
/*start a timer that expires when the packet transmission is complete*/
mhTxPkt_.start(Txpkt_->copy(),txt);
/*pass the packet to the physical layer*/
downtarget_->recv(Txpkt_, this);
…
}

/*To handle incoming packet*/
void TdmaMac::recv(Packet* p, Handler* h) {
    /* Incoming packets from phy layer, send up to ll layer. */
    …
    sendUp(p);
/* Packets coming down from ll layer, send them to phy layer. */
      …
    sendDown(p);
}
```

3.3 Routing Layer

The C++ code for the routing layer is included in the *aeroroute* files. The current routing implementation in Aero-Sim is a centralized routing method, which is implemented entirely in C++. Furthermore, to support distributed routing protocols in AANETs, Aero-Sim makes use of a routing agent at each node. Like the mobility extension parts of NS2, routing packets can be sent to port 255 of each node. Moreover, Aero-Sim provides standard interfaces at almost every network layer to support advanced features of the routing protocols. For example, geolocation-assisted routing protocols can easily obtain location information from the interface of the *AeroNode* object. Thus, Aero-Sim provides a good platform for developing advanced routing protocols.

Instead of computing new routes when the topology changes, the centralized routing algorithm in Aero-Sim computes routes only when there is a packet to send. Furthermore, a single-source shortest-path algorithm is executed instead of an all-pairs shortest path algorithm. The shortest-path route computations use the current propagation delay of a link as the cost metric. Considering that the number of aeronautical nodes is very large, the original centralized routing method would produce a very slow runtime. Thus, we make a major revision to the routing algorithm. Since the aircraft are continuously taking off and landing, to speed up the simulation, when a node computes its adjacency and routing table, it does not consider nodes representing aircraft that have not yet taken off or have already landed. So, the sizes of the adjacency and routing tables of a node are greatly reduced, especially when there are only a few aircraft flying. This yields a run-time performance improvement. This revision involves three aspects, as follows.

First, to maximize the use of the original code, we use an array to store the required nodes. The adjacency table uses the node id as an index into the array to record the link adjacency information.

```
void AeroRouteObject::compute_topology()
{
…
/*insert the airports, satellites and the aircraft that are flying into the array */
air_coord = ((AeroNode* )nodep)->position()->coord();
      if(air_coord.r != 0 ) {
          real_node[realnum] = nodep->address();
          realnum++;
       }
for (nodep=Node::nodehead_.lh_first; nodep; nodep = nodep->nextnode()) {
   for (alhp =(AeroLinkHead*) nodep->linklisthead().lh_first; alhp;
            alhp = (AeroLinkHead*) alhp->nextlinkhead()) {
…
/* src_index and dst_index are the index of the source and destination node */
insert_link(src_index+1, dst_index+1, delay, (void*)alhp);
…
   }
}
```

Second, when the sending node computes its routing table, the node id is the index of this value in the array computed by function compute_topology().The shortest-path algorithm is not changed.

```
void AeroRouteObject::node_compute_routes(int node)
{
...
/* get the node index*/
for(int i=0; i<realnum;i++) {
        if(real_node[i] == node) {
          node_index = i;
          break;
         }
      }
int k = node_index + 1; // add one to get the right offset in tables
/* compute routes only for node "k" */
…
}
```

Finally, when look up the route table to get the next hop nodes and the entry to construct the forwarding table, we also use the index of the node id. The return value of the look up is the index of the next hop nodes, and thus, we must use the index in the real_node[] array to get the real next hop node.

```
void AeroRouteObject::populate_routing_tables(int node)
{
...
next_hop_index = lookup(src_index, dst_index);
next_hop = real_node[next_hop_index];
```

```
…
target = (NsObject*) lookup_entry(src_index, dst_index);
/* construct the forwarding table*/
((AeroNode*)snodep)->ragent()->install(dst, next_hop, target);
...
}
```

4. Case Studies

In this section, we present several case studies to illustrate some of the capabilities of our simulator. By simulating aircraft flying using realistic flight data, we show that Aero-Sim can reproduce the real world with high fidelity. Then we demonstrate the validation of Aero-Sim via an AANET simulation example while the fundamental routing and propagation delay are also analyzed.

4.1 Fidelity Testing

To test the fidelity of Aero-Sim, our first experiment with Aero-Sim was designed to study the possibility of aeronautical ad hoc networking in China's airspace and analyze ground connectivity from the perspectives of the whole network and a single aircraft using realistic domestic flight data.

4.1.1 Data Collection

First, we queried (on the Internet) the average passenger traffic in domestic airports from 2011 to 2013 and ranked the data to get the top 24 busiest domestic airports. Since the traffic in other airports is relatively small, we ignored these airports. Then, we collected detailed inbound and outbound flight information for the 24 airports for a week in October 2013. The flight information changes slightly from one day to the next, and we chose a representative average day for our simulations. Next, we used the latest database in the global route query system provided in AIRCN [16] to query detailed flight routes of the collected flights. This enabled us to obtain the latitude and longitude of the navigation points between the departure and destination airports. For simplicity, we chose navigation points with differences in latitude or longitude no less than three degrees.

4.1.2 Flight Simulation

According to the flight data collected in Section 4.1.1, there are on average 10,521 aircraft in China's airspace on any day. Beijing Capital International Airport, Shanghai Pudong International Airport, and Guangzhou Baiyun International Airport were chosen as ground stations to communicate with aircraft while airborne. Detailed position information of the airports and the flight schedules of all commercial airlines are specified in the file ~ns/tcl/mobility/scene/route-aircraft. Considering that a few aircraft would not have landed by midnight on the simulated day, the simulation time was set to 90,000 s to simulate the entire flight process, with the locations of the aircraft recorded every 3,600 s. **Table 1** shows the main simulation parameters for Aero-Sim, including *R*, which denotes the air–air transmission range. In this experiment, we were interested in the basic characteristics of the dynamic topology created by the AANET, regardless of the routing technique used to forward packets over the network.

Table 1. Parameter configuration for Aero-Sim

Parameter	Parameter Value
Number of aircraft	10521
Number of airports	3
Flight altitude	10 km
MAC type	TDMA
Physical layer	AeroPhy
Air–air transmission range	100 km, 300 km, 600 km
Air–ground transmission range	400 km
CBR packet size	512 bytes
CBR packet interval	50 seconds
Bandwidth of air–air link	10 Mbps
Bandwidth of air–ground link	20 Mbps
Log time of locations	3600 seconds
Simulation time	90000 seconds

4.1.3 Networking Possibility and Connectivity Analysis

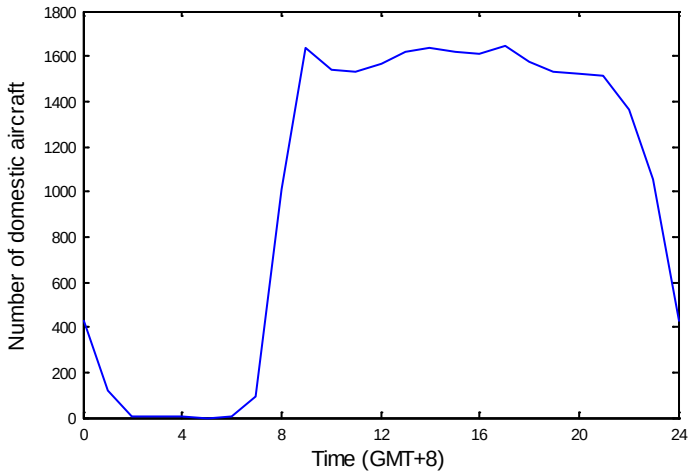

Fig. 6. Variation in the number of aircraft in China's airspace throughout the day

Fig. 6 shows the number of aircraft found within China's airspace throughout the day. The results show that the number of domestic flights in China's airspace varies considerably depending on the time of day. Between 08:00 and 23:00, the number is greater than 1000, with a peak of 1600 during the period 09:00 to 17:00. From 02:00 to 05:00, there are only a few domestic aircraft airborne. **Fig. 7** shows snapshots of the domestic air traffic situation in China's airspace around 01:00, 08:00, and 16:00 GMT+8 on an average day.

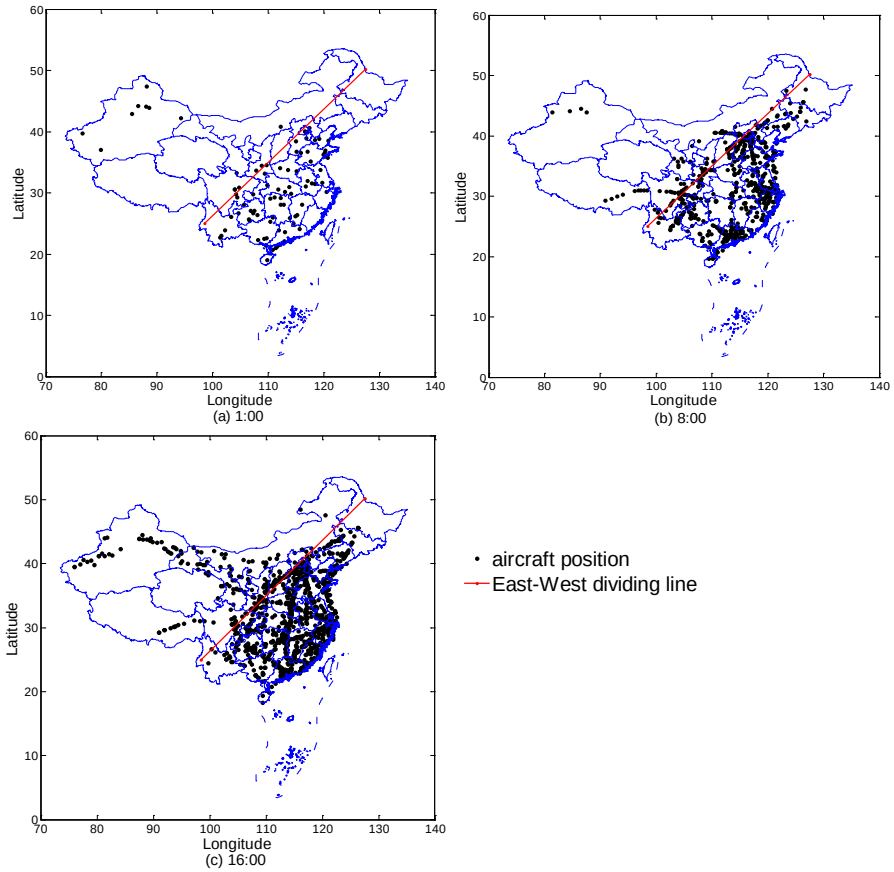

Fig. 7. Snapshots showing aircraft distributions in China's airspace at (a) 01:00, (b) 08:00, and (c) 16:00 GMT+8

The Chinese continent can be divided into Eastern and Western parts by the Moheng–Tengchong dividing line. We analyzed the numbers and densities of aircraft in these two parts. As shown in **Fig. 8** and **Fig. 9**, the number and density of aircraft in the Eastern airspace is much higher than that in the Western airspace. The variation in the number of flights and density throughout the day is very large in the Eastern airspace and relatively smaller in the Western airspace.

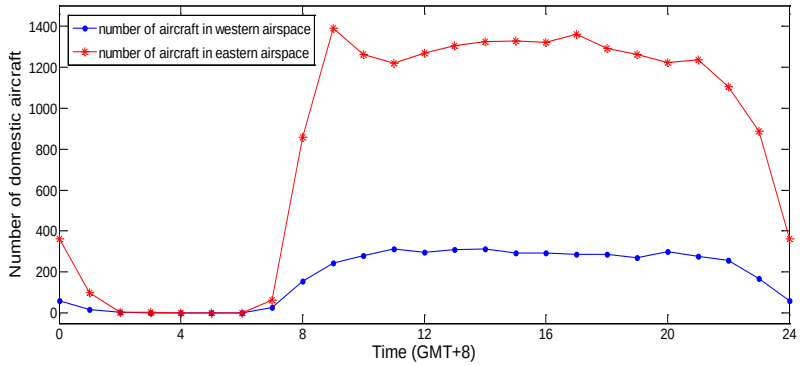

Fig. 8. Variation in the number of aircraft in the Western and Eastern airspace in China throughout the day

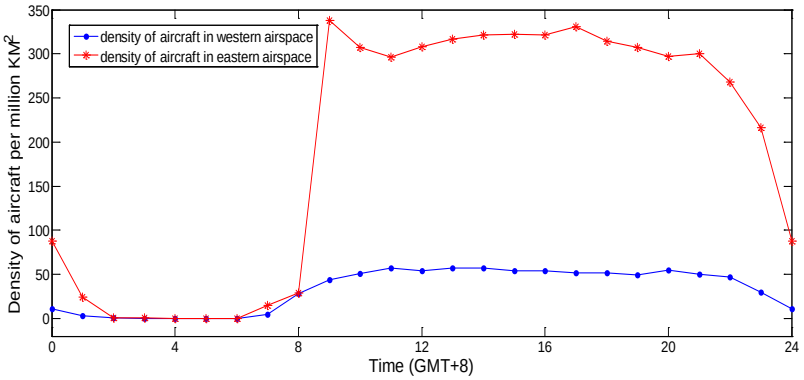

Fig. 9. Variation in the density of aircraft in the Western and Eastern airspace in China throughout the day

Based on the airplane density in the Eastern and Western airspace and the function of networking probability and airplane density [3], we can calculate the probability of aeronautical ad hoc networking in the respective airspaces. As shown in **Fig. 10** and **Fig. 11**, even though the air–air transmission range is 100 km, the probability of aeronautical ad hoc networking between 08:00 and 24:00 in the Eastern airspace is 1. However, in the Western airspace, the communication range must be 300 km to ensure a probability of 1 for aeronautical ad hoc networking between 08:00 and 24:00. Meanwhile, because the number of flights from 01:00 to 07:00 in both the Eastern and Western parts is very small, the probability of aeronautical ad hoc networking during this time is very low even with an ideal communication radius. In other words, it is difficult to establish an AANET during this period.

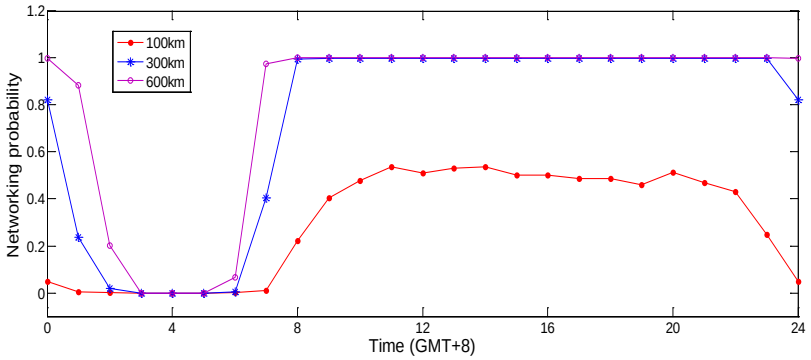

Fig. 10. Networking probability in the Western airspace in China throughout the day

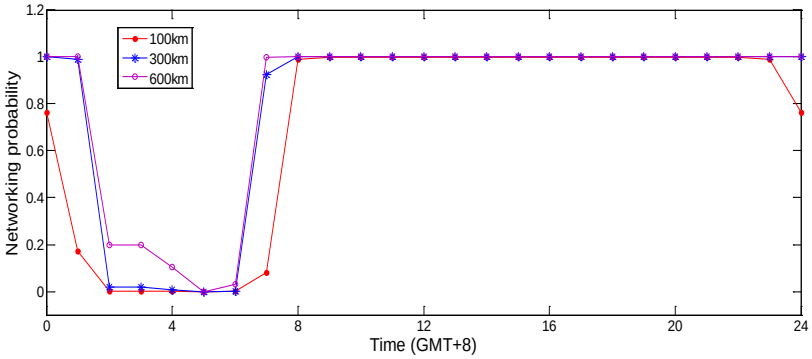

Fig. 11. Networking probability in the Eastern airspace in China throughout the day

Since an AANET provides an effective means for communication between aircraft and the ground stations, the following analysis is aimed at the ground connectivity of the aircraft. The connectivity is usually assessed by the following two metrics [5]:

$$C(t) = \frac{number\ of\ connected\ aircraft\ at\ time\ t}{total\ number\ of\ aircraft\ at\ time\ t} \tag{3}$$

$$\mu = \frac{total\ connected\ time}{flight\ duration} \tag{4}$$

where $C(t)$ is a metric defining the perspective of the entire network. It denotes the fraction of all aircraft that are directly connected or have at least one multiple path to a ground station at a given time. Contrarily, μ is the metric from the perspective of a single aircraft and defines the percentage of the flight duration during which the aircraft can contact a ground station.

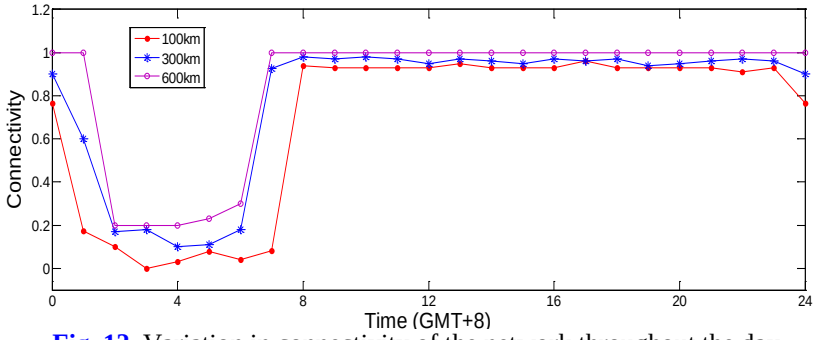

Fig. 12. Variation in connectivity of the network throughout the day

Table 2. Aircraft ground connectivity statistics

R	$\overline{\mu}$ (%)	P (μ >0.99) (%)
100 km	91.334	68.972
300 km	95.996	92.452
600 km	99.507	98.673

Fig. 12 shows the variation in connectivity over 24 hours for several air–air transmission ranges. **Table 2** gives the mean value $\overline{\mu}$, averaged over all flights and the percentage of flights for which $\mu > 0.99$ in the three different ranges. As seen in **Fig. 12** and **Table 2**, ground connectivity of the network improves with an increased air–air transmission range. An air–air transmission range of 600 km is sufficient to guarantee almost 100% connectivity between 08:00 and 24:00. Moreover, in this range, an aircraft is connected to the ground for more than 99% of its flight duration on average. In addition, more than 98% of the flights have permanent connectivity to the ground. In other words, the three ground stations in Beijing, Shanghai, and Guangzhou in China can provide adequate ground connectivity with an air–air transmission range of 600 km.

Considering the experimental results using realistic flight data and the analysis above, it is clear that Aero-Sim can be used to simulate the whole flight process and to investigate networking possibility and ground connectivity, which are the most important problems in AANETs. Furthermore, Aero-Sim closely approximates the results in the real world and thus can be used to reproduce the real world with high fidelity.

Because of the relatively high cost and difficulties of deploying aeronautical nodes, it is often impossible to deploy a real network in every research effort. As an alternative, Aero-Sim provides a powerful tool for carrying out research on AANETs.

4.2 Validation

In this section, we use a simulation example to illustrate how Aero-Sim can be used to implement an AANET. Then, the fundamental routing and delay characteristics are analyzed.

4.2.1 Simulation Setup

In this experiment, we simulate a 1-h time window from 08:00 to 09:00 in the Western airspace in China, corresponding to a typical rush hour on an average day. The airborne network consists of roughly 354 aircraft during this time. These aircraft communicate with the ground stations via an AANET. We concentrated on the aircraft flying from Chengdu to Beijing between 08:00 and 10:20. Constant bit rate (CBR) traffic is sent from the aircraft to the Beijing ground station from 28,800 s to 32,400 s in the flight with a transmission interval of 50 s. Considering that the duration of the inter-aircraft link is relatively long when compared with other ad hoc networks [5], the handoff time of the inter-aircraft link is 100 s. The air–air transmission range is fixed at 600 km. The other main simulation parameters are as given in **Table 1**.

4.2.2 Analysis of Results

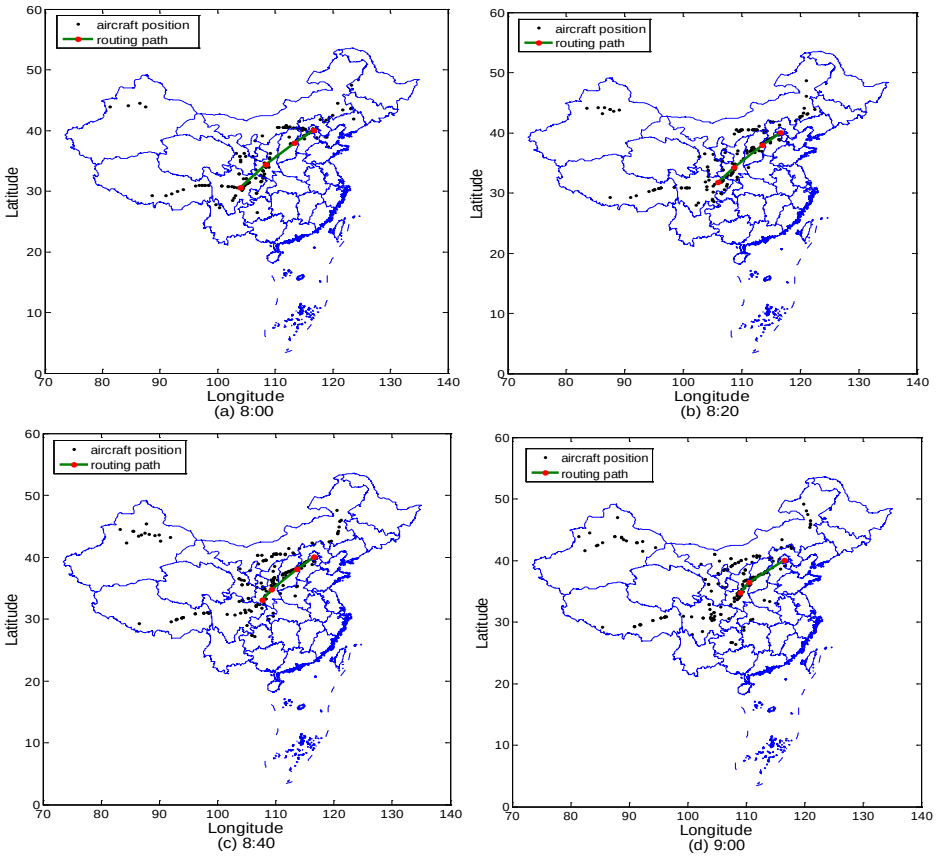

Fig. 13. Routing path from source aircraft to destination Beijing between 28,800 and 32,400 s

Fig. 13 shows the positions of the aircraft flying in the Western airspace, and the paths from the source aircraft to the destination ground station at different times between 08:00 and 09:00. Each source aircraft calculates a routing path every 100 s. Since the aircraft move at high speeds, even if the link status does not change in 100 s, the global min-delay shortest path may be different during this period.

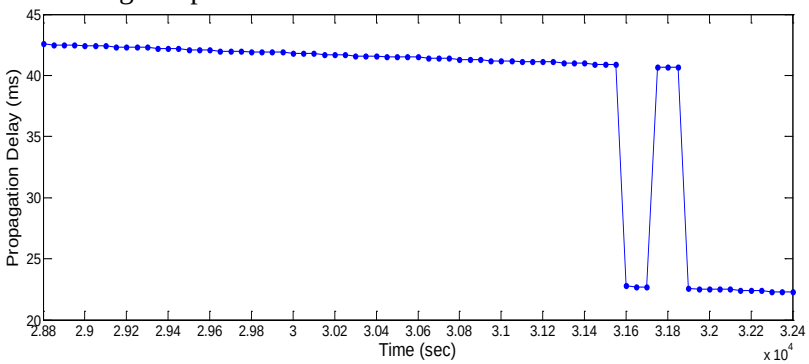

Fig. 14. Propagation delay from source aircraft to destination Beijing from 28,800 to 32,400 s

Fig. 14 plots the end-to-end delay variability between the source aircraft and Beijing ground station from 28,800 s to 32,400 s. The data points represent the delay experienced by a packet sent every 50 s. Generally, as the source aircraft fly towards Beijing, the delay decreases. Moreover, 31,600 s into the flight, the delay is noticeably smaller. This is because at that time, the path between the source and destination only traverses one aircraft hop compared with two hops before that time. However, 31,800 s into the flight, the delay increases sharply because two aircraft hops are required on the path.

5. Conclusion and Future Work

An AANET is a new ad hoc network between airborne commercial aircraft for the purpose of sharing data and Internet access. Owing to the significant distinctions between AANETs and terrestrial wireless networks, a new simulator for AANETs was urgently needed. In this paper, we presented the design and implementation of Aero-Sim, a packet level AANET simulator based on NS-2, which is suitable for simulations of the MAC, link, network, transport, and application layers by composing simulations with existing modules and protocols.

Through several case studies using realistic domestic flight data, we showed that Aero-Sim is a powerful simulation tool with high fidelity that can simulate AANETs accurately. As future work, we intend investigating distributed routing protocols suitable for an AANET and integrating them into Aero-Sim.

References

[1] M. Ilyas, “The handbook of Ad Hoc Wireless Networks,” *CRC press*, 2014. Article (CrossRef Link).

[2] K. Karras, T. Kyristsis, M. Amirfeiz and S. Baiotti, “Aeronautical Mobile Ad Hoc Networks,” in *Proc. of 14th European Wireless Conference*, pp. 3972-3977, June, 2008. Article (CrossRef Link).

[3] E. Sakhaee and A. Jamalipour, “The Global In-Flight Internet,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 9, pp. 1748-1757, September, 2006. Article (CrossRef Link).

[4] E. Sakhaee, A. Jamalipourabd and N. Kato, “Aeronautical Ad Hoc Networks,” in *Proc. of IEEE Wireless Communications and Networking Conference(WCNC)*, vol. 1, pp. 246-251, 2006.

Article (CrossRef Link).
[5] D. Medina, S. Ayaz and F. Hoffmann, "Feasibility of an Aeronautical Mobile Ad Hoc Network Over the North Atlantic Corridor," in *Proc. of 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks(SECON'08)*, pp. 109-116, June, 2008. Article (CrossRef Link).
[6] NS-2. Available at http://www.isi.edu/nsnam/ns/.
[7] OPNET. Available at http://www.opnet.com.
[8] Johnson D B, Broch J, Hu Y C, "The CMU Monarch Project's Wireless and Mobility Extensions to ns," in *Proc. of 42nd Internet Engineering Task Force*, 1998. Article (CrossRef Link).
[9] J. Broch, D.Maltz, D. Johnson, and Y. Hu, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking(Mobicom'98)*, pp. 85-97, October, 1998. Article (CrossRef Link).
[10] T. Henderson, R. Katz, "Network Simulation for LEO Satellite Networks," in *proc. of 18th international communication satellite systems conference*, April, 2000. Article (CrossRef Link).
[11] T. Issariyakul, E, Hossain, "Introduction to network simulator NS2 (Second Edition)," *Springer-Verlag New York Inc.*, 2012. Article (CrossRef Link).
[12] K. Fall, K. Varadhan, "The ns Manual (formerly ns Notes and Documentation)," *The VINT project*, 2005. Article (CrossRef Link).
[13] Ho Dac Tu, Shigeru S. "A proposal of relay data in aeronautical communication for oceanic flight route employing mobile Ad Hoc network," in *proc. of First Asian Conference on Intelligent Information and Database Systems*, pp. 436-441, April, 2009. Article (CrossRef Link).
[14] Ho Dac Tu, Shigeru S. "A proposal for high Air-traffic oceanic flight routes employing Ad Hoc networks," in *proc. of IEEE Wireless Communications and Networking Conference(WCNC'09)*, pp. 1-6, April, 2009. Article (CrossRef Link).
[15] Kleinrock L. and Tobagi, F.A., "Packet Switching in Radio Channels: Part I-Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Transactions on Communications*, vol. 23, no.12, pp. 1400-1416, December, 1975. Article (CrossRef Link).
[16] The global route query system. Available at http://rte.aircn.org/routefinder/index.php

Qin Luo received the M.S. degree in Communication and Information System from University of Electronic Science and Technology of China, Chengdu in 2007. She is currently a Ph.D. student at the School of Aeronautics and Astronautics, Sichuan University. Her research interests cover a wide variety of topics in aeronautical telecommunication networks and spatial information networks.

Junfeng Wang received the M.S. degree in Computer Application Technology from Chongqing University of Posts and Telecommunications, Chongqing in 2001 and Ph.D. degree in Computer Science from University of Electronic Science and Technology of China, Chengdu in 2004. From July 2004 to August 2006, he held a postdoctoral position in Institute of Software, Chinese Academy of Sciences. From August 2006, Dr Wang is with the College of Computer Science, Sichuan University as a professor. His recent research interests include spatial information networks, network and information security, and intelligent transportation system.

Xiaoqing Wang is Professor in the Beijing Institute of System Engineering. He received the Ph.D. degree in Computer Science and technology from Tsinghua University. His technical interests include information management, data sharing, software testing, and software quality assurance. He has rich experience in software research and development project management.

Ke Wu received the B.S. degree in Computer Science and Technology from Sichuan University, Chengdu in 2014. She is currently a M.S. student at the College of Computer Science, Sichuan University. Her research interests include aeronautical networks and interplanetary Internet.