Routing Algorithm for Aeronautical Ad Hoc Networks

Project Proposal

by –
Aditee Nagle
Ronald Bayross
Rucha Gade

COEN 233 - Computer Networks

Winter 2014

Table of Contents

List of Figures

List of Tables

Abstract

Aircraft ad hoc networks (AANET) are being studied to open up the possibility of data communication amongst remote aircrafts without the help of fixed ground infrastructure. High speed of aircrafts, low bandwidth and varying node density pose challenges to routing in AANETs. Geographical routing protocols are suitable for large mobile ad-hoc networks due to scalability and efficiency. In this project, we simulate a modified greedy forwarding geographical routing algorithm to that can be used in AANETs. We compare our results with other routing protocols available in literature.

1. Introduction

1.1 Objective

This project attempts to build an efficient geographical routing algorithm for Aeronautical Ad Hoc Networks (AANETs) and compare the results with latest research work.

1.2 What is the problem

Wireless ad hoc networks with mobile nodes are termed as Mobile Ad Hoc Networks (MANETs). Mobile nodes communicate with each other wirelessly and discover best paths to transfer desired information between source node and destination node. Each node in MANET thus acts as both, a host and a router. Examples of MANETs are people at conferences using their mobile devices to connect with each other, emergency workers at rescue operation site etc.
In Aeronautical Ad Hoc Networks (AANETs), an aircraft becomes the mobile node. As long as the aircrafts are within the communication range of ground stations, they can exchange information directly with the ground infrastructure. But in situations like flights over oceans or hostile regions, where there is no fixed infrastructure available, communication becomes a problem. Using satellite communications is a solution but it is often expensive and has latency problems. Such situations demand for an ad hoc network amongst cruising aircrafts that can pass information hop by hop. Example of AANET can be surveillance drones flying over oceans or forests, aircrafts flying the trans-Atlantic or trans-Pacific routes.
In order to send data from one node (source) to another (destination), the network needs to forward data packets through a series of intermediate nodes. The process of discovering this route becomes complex in case of mobile nodes since there is no fixed route between any two hosts that is always present. Changes in topology may break some of the old routes and give rise to new routes. Thus a robust routing protocol is necessary for efficient exchange of information in MANETs and AANETs.

1.3 Why this is a project related the this class

In the OSI model of computer networks, network layer is situated between the data link layer and transport layer. It transfers packets from the source to their destination through a number of intermediate routers. In order to accomplish its

task, network layer must be aware of the topology of the network. The main functions performed by this layer are:

1. Routing
2. Congestion control
3. Internetworking

The routing process consists of forwarding packets between a source and a destination through network of nodes linked to each other. Routing makes sure that all the packets reach their correct destination with minimum delay. Simplicity, robustness, fairness and stability are desired qualities of any routing protocol.

This project attempts to study and improve one such routing algorithm for the specific application of AANET.

1.4 Why other approach is no good

As with the MANETs, routing in AANETs is complicated due to mobility of nodes. Adding to the complexity, in AANETs, the aircrafts move with much higher speed as compared to nodes in typical MANETs mentioned above. Also, the density of aircrafts cruising in a region changes with time resulting in a large number of mobile nodes at peak times (like day time) and very few nodes during night. Another difference between a typical node in MANET and that in AANET is the restriction on energy consumption. Energy efficiency, though desirable, is not a main concern in AANETs. In recent years, there has been rapid growth in low-altitude flights. Problems of low bandwidth and connectivity are inherent to these flight operations. Due to these and other aspects specific to AANETs, routing protocols used in typical MANETs cannot be applied to AANETs directly.

1.5 Why you think your approach is better

A variety of routing protocols have been developed for MANETs. They can be broadly classified as topology-based and geography-based as shown in Figure 1.

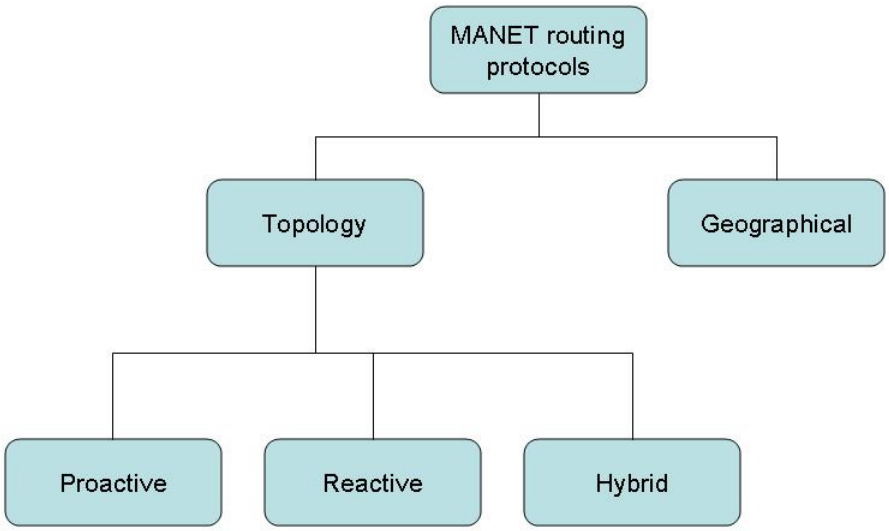


Figure 1: Routing protocols in MANETs

In topology based routing protocols, information regarding updates to the links is used to find a desired path. High overhead for route maintenance, high latency in route discovery, low scalability are some of the limitations faced by topology-based protocols. Geographical routing (GR) protocols make use of readily available location information to find routes. Since physical location of nodes is a known factor in GR, the overheads of routing tables and route maintenance can be avoided. Also there is no need to have a global view of network topology. Due to these reasons, geographical routing algorithms scale better in MANETs.
Geographical routing protocols like GPSR and LAR base their forwarding decisions on current node location alone. For example, in GPSR, the next hop node is selected as the one which is physically closest to the destination. High velocity of aircraft in AANET poses unique challenges to traditional geographical routing methods. In this project we work with a modified GPSR which accounts for location, speed and direction of motion of nodes. A neighbor node that is nearest to the destination and is moving fastest towards destination is selected as the next hop thus reducing the forwarding delay.

1.6 Statement of the problem

To account for high velocities of nodes in AANETs, an improved geographical routing algorithm is required. The algorithm needs to address typical problems involved with AANET like network sparseness, high-mobility, variation in network density and reliability of communication. The algorithm can utilize location and velocity information of nodes available through the ADS-B system that is installed on all new aircrafts.

1.7 Area or scope of investigation

In this project, we focus our attention on following aspects:

1. Decision metrics for data forwarding
2. Comparison with A-GR routing protocol [1]

2. Theoretical bases and literature review

2.1 Definition of the problem

As mentioned in section 1, AANET nodes have high mobility and limited bandwidth, which poses challenges for routing protocols. The connection interval can be short and intermittent. The node density varies with time of flight. A robust and simple routing algorithm is required for AANETs.

2.2 Theoretical background of the problem

Geographical routing protocols use physical location of nodes to make forwarding decisions. The location information is obtained from location services like GPS. Thus it is assumed that every node in the network knows its own coordinates at any given time. Forwarding decisions are based on location information of the destination node and immediate neighbors.

2.2.1 GPSR

GPSR (Greedy Perimeter Stateless Routing) is one of the famous geographical routing protocols. Nodes communicate their location to neighbors through "Hello" beacons. Each node on the route forwards the packet to one of its neighbors, which is physically closest to the destination node. The greedy forwarding proceeds by choosing most efficient next hop. In Figure 2, node x selects y as its next hop to reach destination D since y is x's neighbor closest to D. In the absence of suitable neighbors, the algorithm traces a perimeter around the void area to reach the destination.

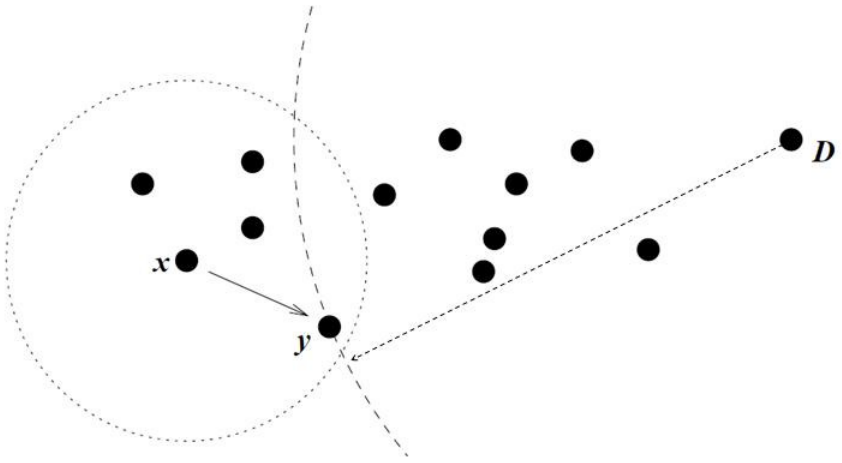


Figure 2: GPSR greedy forwarding

2.2.2 ADS-B

With the advent of new technologies like ADS-B (Automatic Dependent Surveillance-Broadcast) for Air Traffic Management, aircrafts now a days are able to broadcast a lot of information to neighboring aircrafts. A typical ADS-B broadcast includes state vector such as ID, position, velocity, altitude and other flight-related information. This valuable information can aid the routing algorithm and make them more efficient.

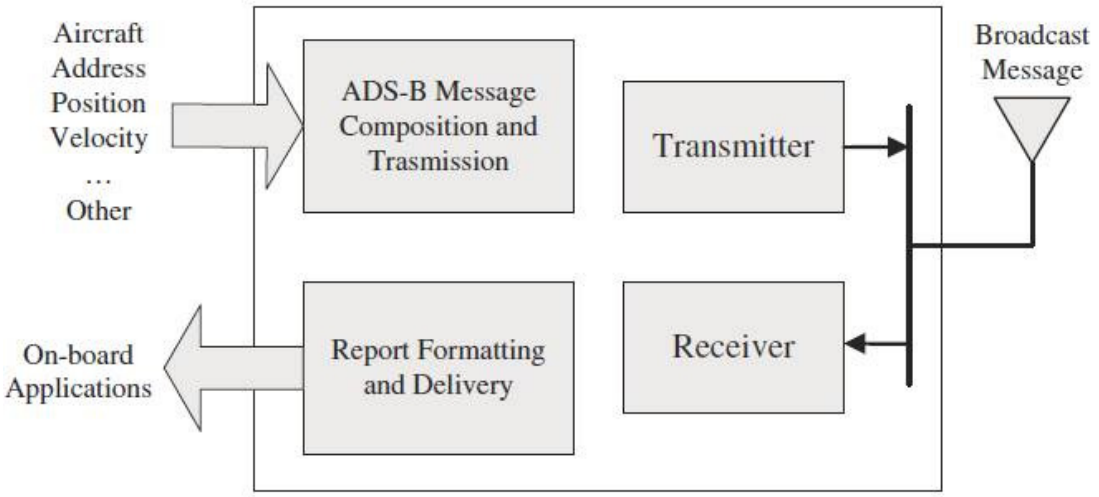


Figure 3: ADS-B on modern aircraft

2.3 Related research to solve the problem

An ADS-B based geographical routing protocol (A-GR) is proposed by Wang et al[1]. It integrates ADS-B into the routing protocol to find the first-hop neighbors. This way, the overhead of Hello beaconing is removed. A-GR uses a decision metric named instantaneous flight time (IFT) to make forwarding decisions. An intermediate node calculates IFT for each of its neighbors and chooses the one with smallest positive value of IFT.

Khan and Ki-Il[2] also use ADS-B information in their geographical routing algorithm (GHRAA). The algorithm uses a modified geographical method to predict second-hop node's location to forward the data.

2.4 Advantage/disadvantage of those research

Both protocols mentioned in section 2.3 are based on traditional geographical routing methods. Both consider the high velocity of moving aircraft when deciding the next hop. In [1] the velocity factor is accounted for by using a metric

called the instantaneous flight time (IFT) which is a ratio of difference in Euclidean distance of source and neighbor from the destination to their relative velocity. The authors claim that the IFT selects a next hop node that will soonest to reach the destination. In [2], the velocity factor is considered by predicting future locations of neighbors and its neighbors. It selects nodes that are most likely moving towards the destination. After establishing a primary path in this manner, an optimized path is discovered using the primary path nodes.
The second-hop prediction in [2] increases due to overheads in routing decisions. Also, the use of primary and secondary path prediction adds complexities. The IFT metric in [2], though simple, leads to selection of a neighbor that is close to the source. The protocols do not explicitly mention the recovery strategy that is applied when forwarding halts due to lack of suitable neighbors.

2.5 Your solution to solve this problem

In this project we plan to use a geographical routing algorithm based on a metric called time-to-reach-destination (TTRD) which is based on GPSR. The time-to-reach-destination (TTRD) metric is calculated for each neighbor node (n) in the following way:

$$TTRD = Dist(n, D) / RV(n, D) \qquad \text{(Eq. 1)}$$

Dist(n,D) = the Euclidean distance between a neighbor node n and destination D
RV(n, D) = relative velocity of n and D along straight line joining n and D
A node having smallest TTRD towards D is the one closest to D and moving fastest towards D, hence it is selected as the next hop.
In the case where there are no neighboring nodes available, the node may keep the packet with itself and make several attempts to forward it after a time interval of Δt. If it fails to forward in these attempts, the packet will be dropped. In case there are no suitable neighbors for next hop available other than a ground station, the message is routed through another ground station that is nearest to the destination.

2.6 Where your solution differs from others

We try to implement a modified metric based on GPSR that makes a greedy decision for packet forwarding. As compared to A-GR protocol, TTRD metric chooses a node which is closer to the destination than source.

2.7 Why your solution is better

Using time-to-reach-destination metric defined in section 2.5 forwards packets fast towards the destination node. Since we use the relative velocity between the node and destination, a node moving towards destination with highest relative velocity gets selected thus reducing the transit time and number of hops.

3. Hypothesis (or goals)

We believe that using a modified greedy forwarding geographical algorithm will reduce the forwarding delays thus making the routing process more efficient.

4. Methodology

4.1 How to generate/collect input data

We have developed a Java-based network simulator to simulate a flight take-off landing schedule. We run the simulator to pass multiple messages between moving nodes and/or ground stations and record the number of successful message deliveries, number of hops and time to deliver for each message. Each test case is run for our algorithm as well as A-GR algorithm to generate data for comparison.

4.2 How to solve the problem

We are designing and implementing a program to provide a way of communicating between aircraft nodes directly or between an aircraft node to the ground station. We have two types of nodes:
1) Moving aircraft nodes with a fixed speed
2) Ground stations that receive and transmit messages between ground stations and between a ground station and an aircraft.
Each node can act as a source, a router or a destination.
When an aircraft wishes to communicate with either another aircraft or a base station, it needs to find the next hop in the direction of the destination. We use a time-to-reach-destination metric explained in Section 2.5. TTRD helps us to find the best possible next hop that makes the most progress towards the destination. This process repeats till we reach the destination. Each node tries to transmit the message forward. If it fails to do so after certain number of attempts, the message is dropped.
In case a flight cannot forward the message to any other flight but a ground station, that station finds another station nearest to destination and routes message that way. Each ground station can communicate with every other ground station directly. In this way, the message may reach the destination in a reverse way.

We also keep track of which nodes participated in the routing of the message originating at the source. At the end we display the number of messages sent by each aircraft and whether it reached the destination.

4.2.1 Algorithm Design

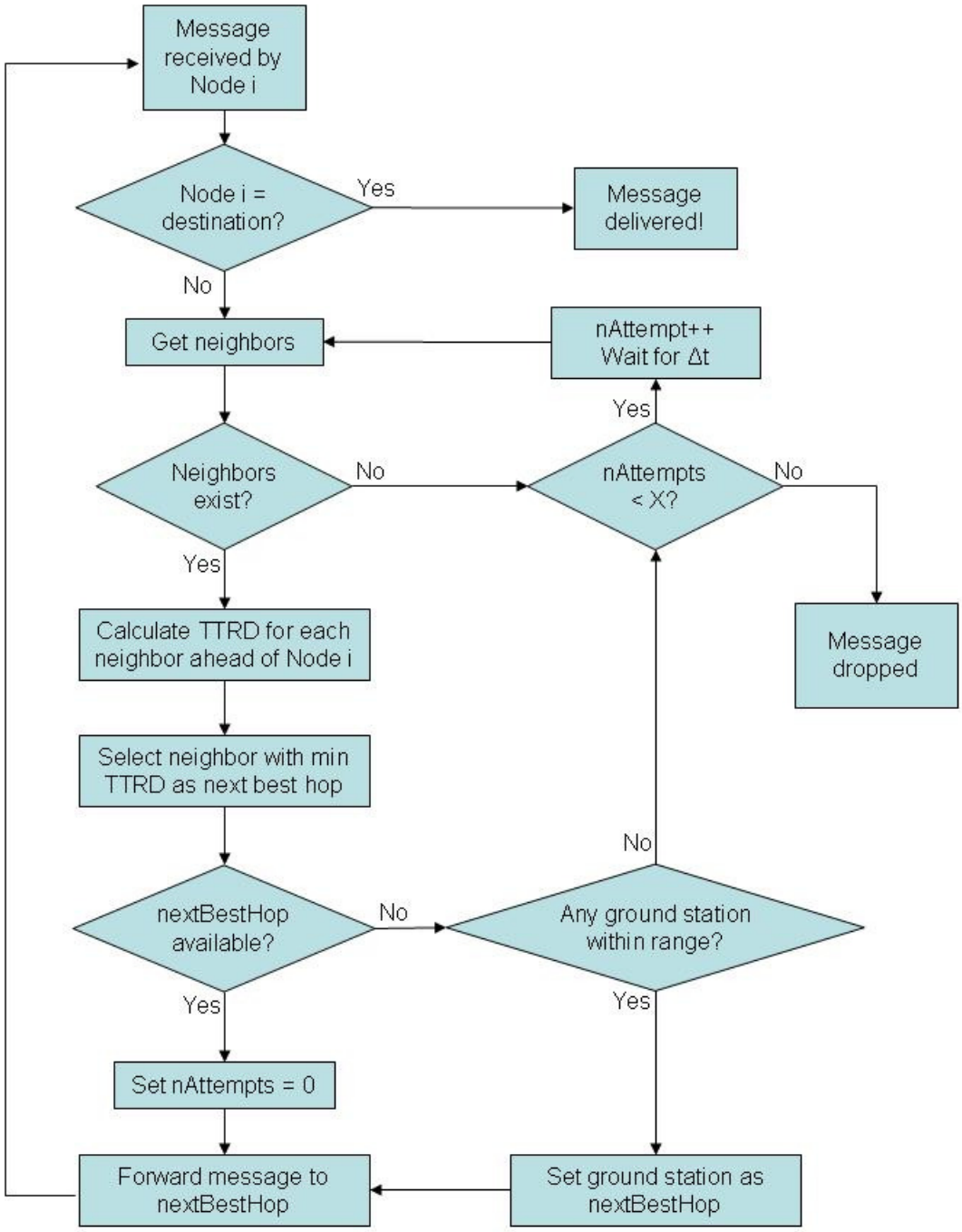


Figure 4: Flow chart

4.2.2 Language used

Java

4.2.3 Tools used

Java SDK7, NetBeans and Eclipse IDE.

5. Implementation

5.1 Simulation

5.1.1 Simulation layout

We have developed a Java based simulator that has an aircraft mobility generator, node objects that can route message packets to their neighbors and a separate messaging thread that sends messages to nodes for routing. Simulation layout is shown in Figure 5.

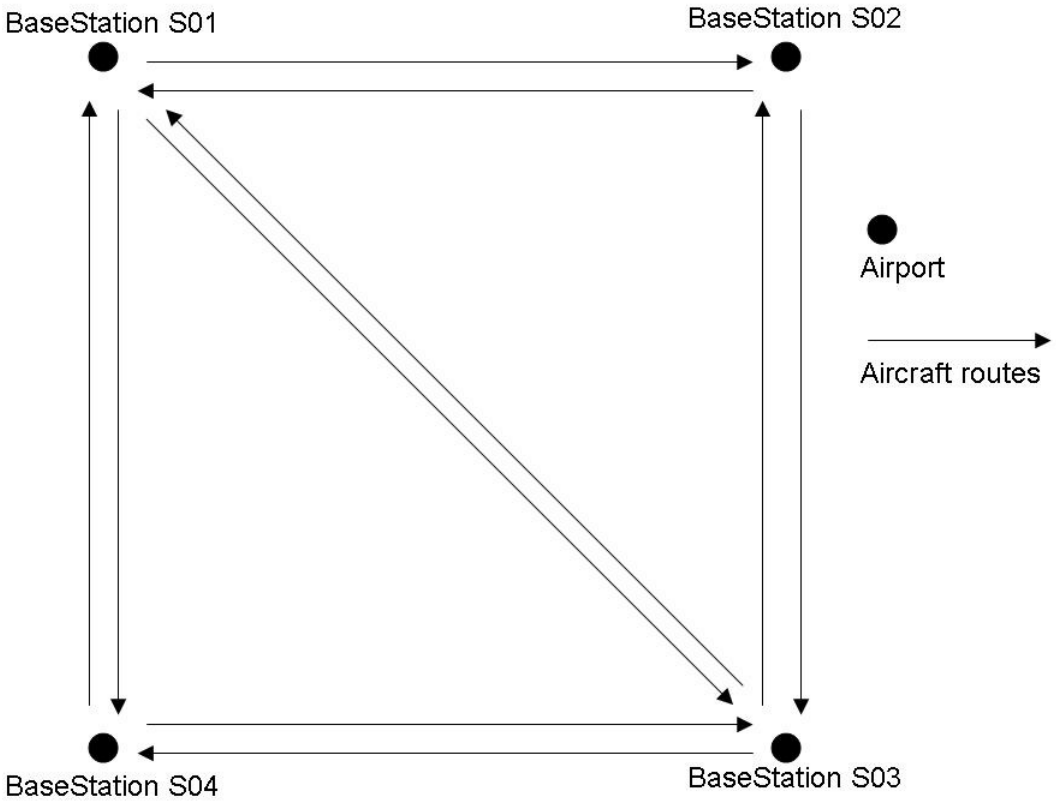


Figure 5: Simulation layout

Flights move between the airports along specified routes. Each airport has a different frequency of flights taking off. Flights and aircrafts can communicate with each other when they are in communication range (R). Mobility manager of simulator updates node locations from time to time. Nodes maintain their own neighbor list (other nodes within communication range) which gets updated every time new location data is available. The messaging thread creates and sends messages to either aircrafts which are in flight or base stations to forward them to a certain destination. Destination can be an aircraft in flight or a ground station. Receiver node then routes the message to one of its neighbors based on TTRD metric. The parameters used for simulation are given in Table 1.

Table 1: Simulation parameters

Network field size	4000 km x 4000 km
Aircraft speed	200 to 250 m/s
Aircraft altitude	20000 to 30000 ft
Communication range	500 km
Simulated time	12 hours

5.1.2 Assumptions

For this study, we have made following assumptions to simplify the mobility models and communications:

- Flights do not collide while landing at the airport. The Air Traffic Control (ATC) controls the take-off and landing schedule at airports to avoid collision. In our model, we have not included those controls.
- Flights and ground stations have same communication range.
- Flights maintain constant speed and altitude during flight.
- Only flights in air can communicate with each other and ground stations. Once a flight is landed, it cannot participate in routing.

5.2 Code

Our simulator program is a multithreaded Java application. It is organized as shown in Figure 6.

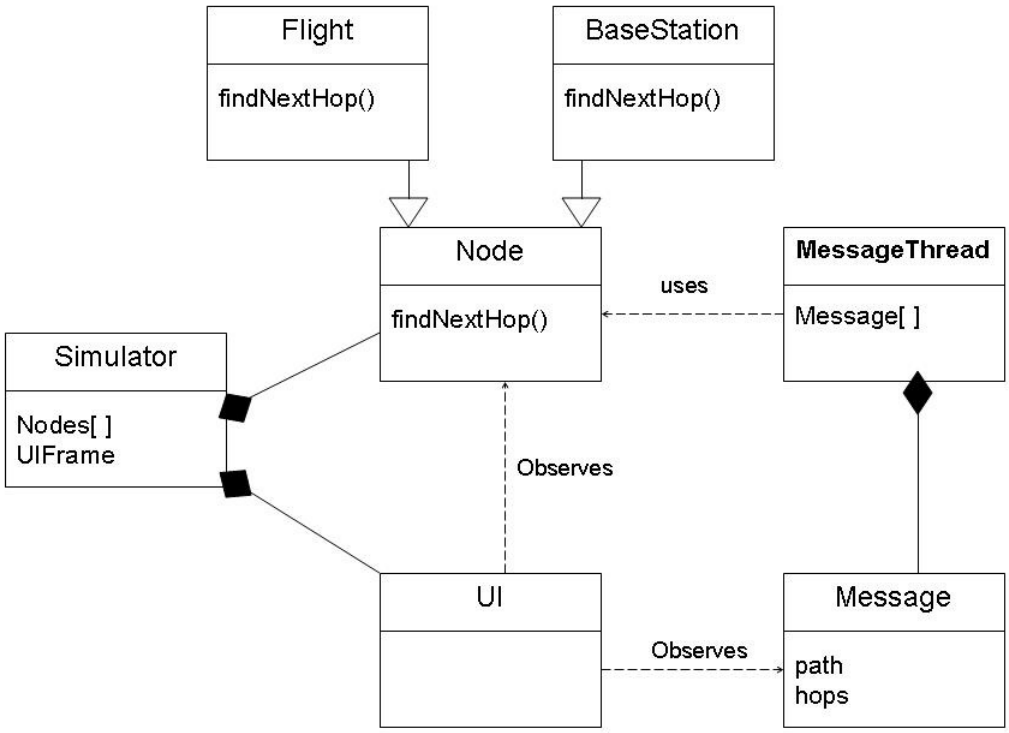


Figure 6: Class Diagram for Simulation App

1. Simulator class: Contains main method which keeps simulation time and updates node location at every new time point.
2. Node class: This is a super-class containing common members and methods belonging to any node. Node could be a ground-station or an aircraft.
3. Flight class: This is a subclass of Node. An aircraft node is defined as a Flight.
4. BaseStation class: A ground-station is defined as BaseStation and is sub-class of Node.
5. UI class: This class shows graphic map of moving/stationary nodes.
6. Messaging Thread class: This is a thread that generates messages and passes them to source nodes for routing.
7. Message class: This class stores message-specific details like number of hops, nodes in path, message delivery success status etc.

Each node finds a next best hop amongst it own neighbors when a message is handed over to it. Classes Node, Flight and BaseStation contain method findNextBestHop() to do this routing.

5.2.1 Sample code for finding next hop using TTRD

A snippets of code from *findNextBestHop( )* method in Flight Class is given below.

Step 1: Finding distance between neighbor node and destination node

```
//Get coordinates of neighbor
x1 = n1.getXCoordinate() * 1000; // Converted Km to m
y1 = n1.getYCoordinate() * 1000; // Converted Km to m
z1 = n1.getZCoordinate() * 0.3048; // Converted feet to m

//Get coordinates of destination
x2 = dest.getXCoordinate() * 1000; // Converted Km to m
y2 = dest.getYCoordinate() * 1000; // Converted Km to m
z2 = dest.getZCoordinate() * 0.3048; // Converted feet to m

//Calculate Euclidean distance netween neighbor and destination
dist = Math.sqrt(((x2 - x1) * (x2 - x1))
                              + ((y2 - y1) * (y2 - y1))
                                          + ((z2 - z1) * (z2 - z1)));
```

Step 2: Finding relative velocity between neighbor node and destination node. Relative velocity of neighbor node and destination node is calculated subtracting x,y,z velocity components of neighbor from those of destination. Component of this relative velocity along the line joining these two nodes is then calculated as dot product of two vectors divided by absolute value of first.

```
//Get velocity vector of neighbor
if(n1.getClass().getSimpleName().equals("Flight")) {
      if (((Flight) n1).getFlightStatus().equals("INFLIGHT")) {
            vx1 = ((Flight) n1).getVelocityX();
            vy1 = ((Flight) n1).getVelocityY();
            vz1 = ((Flight) n1).getVelocityZ();
      }
```

```
}
else {
      vx1 = 0;
      vy1 = 0;
      vz1 = 0;
}

//Get velocity vector of destination
if(dest.getClass().getSimpleName().equals("Flight")) {
      if(((Flight) dest).getFlightStatus().equals("INFLIGHT")) {
            vx2 = ((Flight) dest).getVelocityX();
            vy2 = ((Flight) dest).getVelocityY();
            vz2 = ((Flight) dest).getVelocityZ();
      }
}
else {
      vx2 = 0;
      vy2 = 0;
      vz2 = 0;
}

//Calculate relative velocity between neighbor and destination
rel_vel = ((vx2 - vx1) * (x2 - x1) + (vy2 - vy1) * (y2 - y1) + (vz2 -
vz1) * (z2 - z1)) / (dist*1000);
```

Step 3: Calculating TTRD

```
if(rel_vel != 0) {
      ttrd = dist / rel_vel;
}
```

Step 4: Selecting next best hop

For this step, `ttrd` for each neighbor is compared with earlier saved value of best `ttrd` (minimum `ttrd` if `ttrd` > 0 or maximum `ttrd` if `ttrd` < 0). If `ttrd` of current neighbor is better than previous value, current neighbor is selected next best hop and best `ttrd` value is updated.

6. Data Analysis and discussion

6.1 Output generation

Simulator was run for several test cases and data was generated using findNextBestHop function given in Section 5.2. The same test cases were run using IFT metric of A-GR[1] protocol to generate data for comparative study. Design of experiments for this study is given in Table 2.

Table 2: DoE for data generation

Variables	Values
Routing algorithm	TTRD based, IFT based
nAttempts	3, 7, 10

nAttempts is the number of attempts a node can make before dropping the message in case no suitable neighbor is available for forwarding. Output generated from a simulation includes:

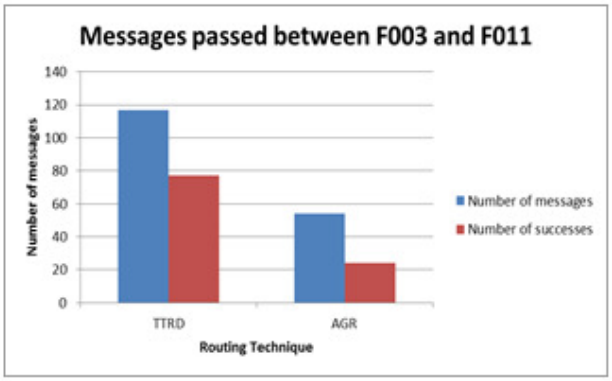
- Total number of messages sent
- Number of messages successfully delivered.
- Total time for message delivery.
- Number of hops taken before either message delivery or message drop
- Path taken by the message

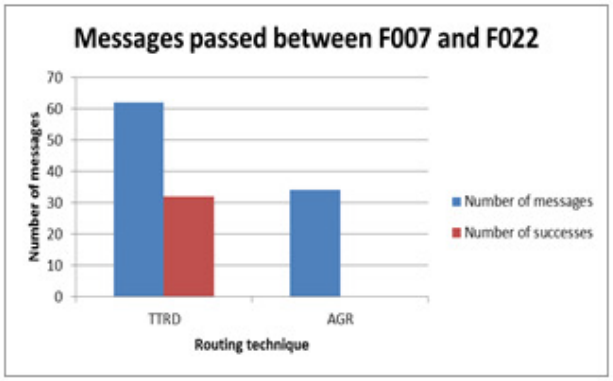
6.2 Output analysis and comparison

Comparison plots for few test cases are shown in Figures to .

Performance of our routing algorithm was compared with IFT based A-GR protocol for:
1. Number of messages generated
2. Number of successful message deliveries

Figures 7, 8 and 9 show results comparison for TTRD vs A-GR for a pair of source and destination for nAttempts equal to 3, 7 and 10 respectively. TTRD based routing algorithm shows high number of messages generated and delivered as compared to A-GR for all three cases.

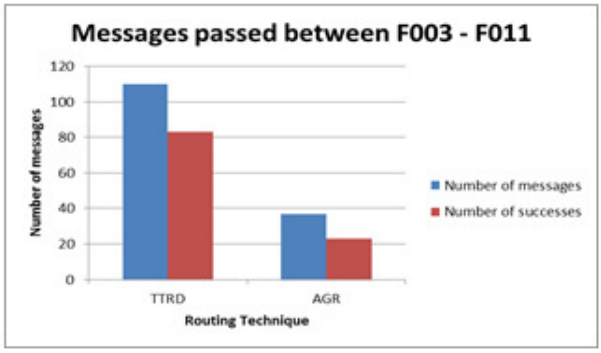


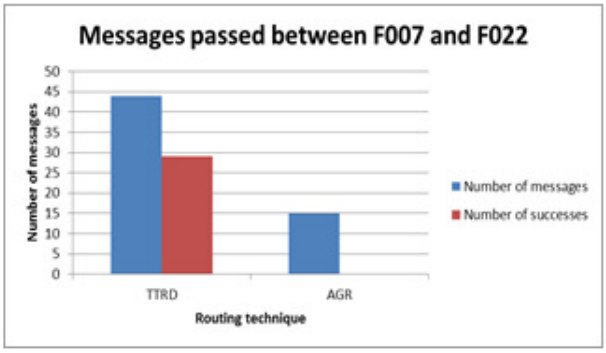


Method	Number of messages	Number of successes	Percentage success
TTRD	117	77	65.81
AGR	54	24	44.44

Method	Number of messages	Number of successes	Percentage success
TTRD	62	32	51.61
AGR	34	0	0

Figure 7: Comparison for nAttempts = 3

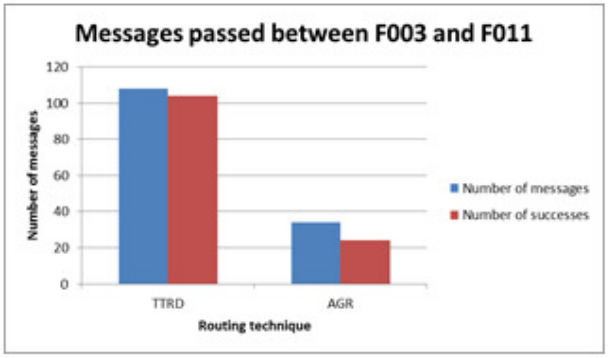


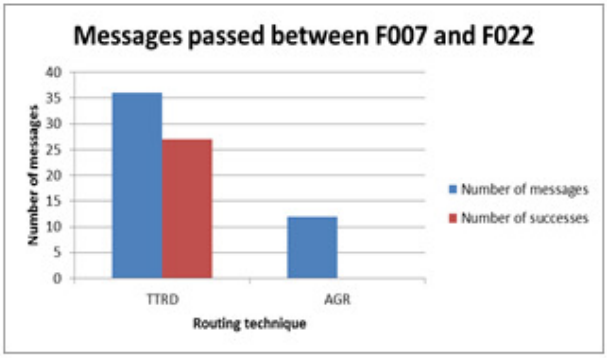


Method	Number of messages	Number of successes	Percentage success
TTRD	110	83	75.45
AGR	37	23	62.16

Method	Number of messages	Number of successes	Percentage success
TTRD	44	29	65.91
AGR	15	0	0

Figure 8: Comparison for nAttempts = 7





Method	Number of messages	Number of successes	Percentage success
TTRD	108	104	96.30
AGR	34	24	70.59

Method	Number of messages	Number of successes	Percentage success
TTRD	36	27	75
AGR	12	0	0

Figure 9: Comparison for nAttempts = 10

The number of messages delivered improved with more nAttempts since a node gets more time to route the packet before it has to give up. Since the current model creates a new message only after a previous one is either delivered or dropped, number of messages created can point to the efficiency of message delivery. TTRD based approach shows more messages created compared to A-GR.

7. Conclusions and recommendations

7.1 Summary and conclusions

In summary, we have developed and used a java based simulator to simulate Aircraft Ad-hoc Network scenario. We tested our TTRD based routing algorithm for several cases and compared our results with those for A-GR, an IFT based geographical routing protocol for AANETs[1]. The comparison study shows that the number of successful message delivery improved with TTRD-based algorithm.

7.2 Future scope

In future, we would like to allow multiple parallel messages being passed between the various nodes in the network to mirror real-life scenario. We would also like to test the algorithm in realistic simulation environment with signal attenuation and network protocols.

8. References

[1] Shangguang Wang, Cunqun Fan, Cao Deng, Wenzhe Gu, Qibo Sun, Fangchun Yang, "A-GR: A novel geographical routing protocol for AANETs", Journal of Systems Architecture 59 (2013)

[2] Khan Saifullah and Ki-Il Kim "A new geographical routing protocol for heterogeneous aircraft ad hoc networks", Digital Avionics Systems Conference (DASC), 2012 IEEE