UAV-Enabled Extension of Wireless Connectivity

William F. Perry and Jeremy A. Hansen
Norwich University
Northfield, VT 05663 USA
Email: wperry@stu.norwich.edu, jeremyhansen@acm.org

***Abstract*—Given the rise of disasters, emergency situations, and non-conventional conflicts around the world, there is a need for a more flexible method of communication for these environments. One solution is the implementation of a Mobile Ad-hoc NETwork (MANET) on an Unmanned Aerial Vehicle (UAV) platform. Though specialized equipment could be developed to fit exact situations, the use of off-the-shelf equipment and easily-accessible software would allow more organizations to implement this technology and make the replacement or expansion of existing MANETs easier. As current routing protocols used on the Internet do not adequately fulfill the needs of a MANET, the Optimized Link State Routing (OLSR) protocol was chosen to manage such a network.**

I. Introduction

Recent disasters have shown the vast gap between technology and the tools needed to quickly and efficiently address emergency situations. Already challenging responses are exacerbated by communication challenges such as where to send medical personnel, water, food, and other resources to be most effective. So far this challenge has been addressed via a number of different avenues: cell phones, satellites and ad-hoc mobile networks set up by responders on scene. These solutions can be affected by local conditions and are not easily adaptable to rapidly changing situations [12], [13]. Mobile networks set up by respondes are the most flexible in changing situations. Many are carried on vehicles that move with the responders' base camp. While mobile networks do a good job of effectively solving much of the communication problem, they do have their drawbacks. These drawbacks are most obvious in difficult terrain such as in New Orleans after Hurricane Katrina or in the aftermath of the 2010 earthquake in Chile. Because situations like these impede free movement, responders have to move slowly and carefully. The advantages of having an effective communication network and the inability of land-based vehicles to maneuver freely in these environments warrants a more effective solution.

We propose the cost-effective approach of Unmanned Aerial Vehicles (UAVs) that provide a Mobile Ad-hoc Network (MANET). MANETs are a continuing area of interest for a large number of people and organizations. The Defense Advanced Research Projects Agency [16], the University of Colorado [9] and the Swiss Federal Institute of Technology [8], have researched methods for implementing MANETs.

MANETs are wireless networks that have the ability to rearrange their topology with no manual reconfiguration. The physical area covered by a MANET is dependent solely on the range of each node's (which could be a personal computer, mobile device, and so on) wireless transceiver and the layout of the nodes with respect to each other. A node within a MANET can move anywhere inside the three-dimensional space covered by the MANET's other nodes without losing connectivity. Because connections between nodes appear and disappear frequently, individual nodes must act as routers for other nodes to maintain the integrity of the MANET.

Our solution is essentially a remote controlled or self-guided aerial platform, like a helicopter or small fixed wing vehicle, with a powerful wireless router on board [11]. Using this platform to deliver both Internet connectivity and the ability to pass data between aerial and ground-based hosts would provide reliable data communications. Responders would be able to talk to each other in real time with video or voice calls. Video calls could prove useful in the event that a responder needed to contact a qualified medical professional to diagnose and treat an injured victim.

Another advantage of using a UAV-enabled MANET is the ability to rapidly change the coverage area of the network to respond to changing circumstances on the ground [10]. This ability to rapidly respond to changing conditions is one of the greatest arguments for this solution. Often, more harm results because responders were not aware of or could not appropriately respond to a newly-developing situation.

II. Concept

Using an off-the-shelf quadricopter, our intent was to show that UAV-enabled MANETs can be easily and cheaply set up by untrained parties such as a small volunteer fire department or local police force. In addition to flexible converage and inexpensive design, being user-friendly, especially to the most basic of users, is one of the main goals of this project. solutions that are overly technical or difficult to set up and use would hinder widespread deployment.

The choice of a quadricopter over a fixed-wing or single-rotor helicopter was based on the quadricopters stability in moderate to harsh wind conditions. The quadricopter is also small and agile enough to allow additional off-the-shelf products to be mounted onto it[1]. In an actual deployment, the choice of which platform to use would depend more on the funds and resources available and the role for which the MANET would be used.

[1]We also suspect that using a fixed-wing UAV would have increased the power needs of the wireless transceiver and the cost of the on-board equipment due to the higher altitude compared to the relatively stationary quadricopter.

Fig. 1. The AR.Drone in flight with the Raspberry Pi MANET node attached.

III. Hardware

The AR.Drone 2.0 [5] quadricopter was selected due to its availability at a number of retailers, its ease of use, the cost, and the capabilities of the craft. The AR.Drone is a four-rotor quadricopter kept aloft by four brushless motors and powered by a lithium ion rechargeable battery. The frame is made of high grade foam with four carbon fiber tubes connecting the motors to the body. Onboard is a gyroscope, 720p high-definition camera, 3-axis magnetometer, and a small ARM-based processor running a modified distribution of Linux. Integrated into the motherboard is an 802.11b/g/n transceiver from which the quadricopter is both controlled and transmits a video feed.

Originally, the AR.Drone was to be the aerial platform as well as to handle the network routing. However, due to unforeseen difficulties, another piece of hardware was used in order to implement the MANET. While the AR.Drone still played a vital role in the project, the bulk of computing fell to another lightweight, easily configurable piece of hardware, a Raspberry Pi [6]. In order to stay within the lift capacity of the quadricopter, it was decided that a Raspberry Pi Model B would be mounted to the AR.Drone to provide the infrastructure of the MANET. The Pi would be tucked into the quadricopters exterior shell and draw its power from the AR.Drone via its USB port.

The Raspberry Pi is a credit card-sized computer with an ARM-based processor, two USB 2.0 interfaces, an RJ-45 Local Area Network (LAN) interface, in addition to other interfaces not relevant to this project. The computer draws its power from a 5V micro USB interface. In order for the Pi to route wireless traffic while being mounted on the AR.Drone, two WiFi transceivers were plugged into the USB interfaces. The Pi supports several different distributions of Linux installed onto an SD memory card. The distribution we selected was Raspbian, in part because of its support for a wide variety of external hardware.

IV. Software

Several possible protocols for MANETs have been proposed, with several gaining more support than others, but with no clear winner. One of these solutions is the Optimized Link State Routing protocol (OLSR) [4].

OLSR is a robust, "proactive" protocol that solves many of the problems inherent with a MANET. It is proactive in that, unlike what is encountered in typical non-mobile networks where, for example, OSPF (Open Shortest Path First) [15] and RIP (Routing Information Protocol) [14] wait a set interval to "discover" routes throughout the network, OLSR instead sends a `HELLO` message out in a much shorter interval; the default being five seconds. This `HELLO` message functions in somewhat the same way as the messages in RIP and OSPF in that it helps OLSR establish which links are available and unavailable. This is important because, unlike with RIP and OSPF, OLSR does not try to create a "map" of the entire network. OLSR keeps track of which other network nodes it can talk to and which nodes its neighbors can talk to, but no further. This dramatically reduces overhead and complexity when the MANET gets large. OLSR hands traffic off to the directly-connected node in the network with the most reliable or fastest connection as opposed to calculating the fastest route through the entire network. This can lead to traffic taking longer than in a network where RIP or OSPF are used. However, in a rapidly changing MANET where connections may or may not be available, the protocol allows traffic to move in an intelligent way.

V. Implementation

OLSR has been implemented on most operating systems and requires only minimal configuration. The AR.Drone runs a version of Linux capable of the minimum functionality to fly the quadricopter. Our original plan was to install OLSR on the quadricopter without adding any additional hardware, as described in [2]. Because the distribution of Linux on the AR.Drone only provided a minimal set of software, the resources to fetch, configure and install software packages were not present. This made the first step of installing OLSR directly on the onboard computer impossible. We chose instead to use a cross compiler to produce a binary version of OLSR compatible with the quadricopter's hardware.

Further technical difficulties arose during our attempt to build the binary version of OLSR. Though the majority of the tools and source code resources needed to cross compile OLSR for the quadricopter were available, there were a crucial handful that were not or were available only in incompatible versions.[2] Attempts to use an alternative cross compiler [1] failed for similar reasons. As a result, we had no other option but to look for an alternative to directly running OLSR the AR.Drone.

[2] In particular, the `apt-cross` cross compiler, `gcc-4.2 binutils` and `g++-4.2 binutils`. The latter two could be installed on both the Debian and Ubuntu Linux distributions, but when run indicated that the packages were broken.

Our workable solution was to use the Raspberry Pi, in part because its size and weight allowed it to be mounted to the quadricopter. Running OLSR on the Pi eliminated the need to cross compile, since it was powerful enough and had the resources to be able to configure and install the software on its own. To get OLSR working properly, we installed and configured the two WiFi transceivers, one to participate in the MANET, and the other for management. Alternatively, the second WiFi interface could be used to bridge the OLSR traffic to a local–presumably ground-based–network.

This bridging requires the Pi to communicate with a router that can hand off traffic to and from the MANET and the local (conventional) network. For the purposes of this project, we expected a personal computer or a wireless router to act as the "next hop" router from the MANET. The PC or router participates in the MANET, using OLSR on a wireless interface, and passes traffic to a wired interface.

After some issues using both a wireless router and a PC as the local network-to-MANET router[3], we eventually decided, for the sake of time and future projects, that the Pi should act as its own router. That way, each quadricopter could opportunistically connect to a nearby network and act as a gateway for the rest of the MANET. This adds another layer of flexibility that would not otherwise be possible if users were required to use a dedicated PC or wireless router.

The two WiFi transceivers were plugged into the Pi with one configured to run OLSR and the other connected to a home network. Using `iptables`, traffic was forwarded between the interface running OLSR and the interface connected to the non-OLSR WiFi network. To keep the configuration simple, we set up Network Address Translation (NAT) between the two interfaces. This allowed traffic originating on the MANET to be forwarded to the conventional WiFi network, but not new connections originating from the conventional network.[4]

When a node in the MANET requests an Internet resource, (e.g. a web browser sending an HTTP request to `www.google.com`), the traffic is forwarded through the MANET until it is received by a node with a gateway to the Internet. This could result in substantial congestion and latency in a busy MANET with only one node with Internet access. However, with three or four nodes connected to the Internet distributed throughout or around the edges of the MANET, the latency would greatly decrease and the throughput would increase.

VI. Conclusion and Future Work

Though we were unable to get the prototype fully functional, we can conclude that using a Raspberry Pi attached to a quadricopter to run OLSR and provide flexible network communications is possible.

The Raspberry Pi Model A, if powered by four AA batteries, can run for about fifteen hours [7]. Since the Raspberry Pi Model B draws about twice as much power than the Model A, we expect it to run for more than eight hours. The AR.Drone has a flight time of between 21 and 30 minutes, a small fraction of the time that the Pi can operate. While the AR.Drone cannot fly well with the Raspberry Pi and four AA batteries, the AR.Drone is powerful enough to fly with only the Pi attached. The Pi can then be powered via the USB port on the AR.Drone. This configuration will reduce the overall flight time of the Drone but will run significantly longer than with the AA batteries attached.

We tested the flight time of the AR.Drone with the Pi attached. This adds roughly 60 grams[5] of weight, which cuts the flight time of the AR.Drone to roughly 9 minutes. It should additionally be noted that control of the Ar.Drone was severely impacted with the mounted Raspberry Pi.

With a price tag of just under $400 for the AR.Drone and all of the attached equipment[6], even a resource-constrained organization could consider deploying this inexpensive platform to build a light-duty MANET of their own. We realize that the AR.Drone is underpowered for this purpose and inappropriate for extended use, but the Raspberry Pi could be mounted to almost any other aerial or ground-based platform. Paired with a GPS receiver, a MANET-equipped UAV could be configured to fly preset routes, relieving the need for constant user interaction.

While we created a MANET-equipped UAV as a proof of concept, it was a minimal setup. Our next step is to expand the MANET to include several AR.Drones (each carrying a node) and several ground-based nodes. We would eventually like to explore low-cost replacements for the AR.Drone that would have longer flight times. Much farther down the road, we expect that solar-powered planes [3] would be an ideal platform to carry a MANET node, though such vehicles are certainly out of the price range for most organizations.

References

[1] Available ARM EABI G++ Lite Builds for Mac OS X. http://www.carlson-minot.com/available-arm-eabi-g-lite-builds-for-mac-os-x.

[2] Funkfeuer wiki: Ardrone. http://wiki.funkfeuer.at/index.php/ARDrone.

[3] Helios prototype. http://www.nasa.gov/centers/dryden/news/ResearchUpdate/Helios/.

[4] Optimized Link State Routing Protocol. http://olsr.org/.

[5] Parrot AR.Drone 2.0. http://ardrone2.parrot.com/.

[6] Raspberry Pi. http://www.raspberrypi.org/.

[7] Running the raspberry pi on batteries, author=Dave Akerman, howpublished=http://www.daveakerman.com/?page_id=1294.

[8] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self organization in mobile ad hoc networks: the approach of terminodes. *Communications Magazine, IEEE*, 39(6):166–174, 2001.

[9] T. X. Brown, B. Argrow, C. Dixon, S. Doshi, R. george Thekkekunnel, and D. Henkel. Ad hoc uav ground network (augnet. In *In Proc. AIAA 3rd Unmanned Unlimited Technical Conference*, 2004.

[10] T. X. Brown, S. Doshi, S. Jadhav, and J. Himmelstein. Test bed for a wireless network on small uavs. In *In Proc. AIAA 3rd Unmanned Unlimited Technical Conference*, pages 20–23, 2004.

[3]On the first try, this did not work so well. While attempting to install new firmware onto a wireless router, there was a problem and the router was "bricked" or left inoperable.

[4]In order to do this properly in the future, the Pi would be configured as a router, not just a NAT gateway.

[5]45g for the Raspberry Pi, 12g for the two WiFi transceivers, and 3g for the SD card.

[6]Roughly, $45 for the Raspberry Pi, $40 for the two WiFi transceivers, $5 for the 4G SD card, and $300 for the AR.Drone.

[11] O. Cetin and I. Zagli. Continuous airborne communication relay approach using unmanned aerial vehicles. *Journal of Intelligent and Robotic Systems*, 65(1-4):549–562, 2012.
[12] M. Douglas. Out of thin air. http://urgentcomm.com/mag/radio_thin_air.
[13] W. Guo, X. Huang, and Y. Liu. Dynamic relay deployment for disaster area wireless networks. *Wirel. Commun. Mob. Comput.*, 10(9):1238–1252, Sept. 2010.
[14] G. S. Malkin. RIP version 2 (RFC 2453). 1998.
[15] J. Moy. OSPF version 2 (RFC 2178). 1997.
[16] M. Pinkney, D. Hampel, and S. DiPierro. Unmanned aerial vehicle (uav) communications relay. In *Military Communications Conference, 1996. MILCOM '96, Conference Proceedings, IEEE*, volume 1, pages 47 –51 vol.1, oct 1996.