2013

Development of Mobile Ad-Hoc Network for Collaborative Unmanned Aerial Vehicles

Siva Teja Patibandla
Virginia Commonwealth University

Development of Mobile Ad-Hoc Network for Collaborative Unmanned Aerial Vehicles

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University

By

Siva Teja Patibandla

Director: Dr. Robert H. Klenke

Associate Professor of Electrical and Computer Engineering

Virginia Commonwealth University

Richmond, Virginia

Aug 2013

Acknowledgements

I would like to thank all the people that helped in progress and completion of this work. Specifically I would like to thank my advisor, Dr. Klenke, and my committee members, Dr. Ghosh and Dr. He, for their help and support. I would also like to thank the faculty of Engineering Department for their help and support throughout my years at VCU. I would like to express my gratitude to School of Engineering for funding my study at VCU. I would also like to thank Tim Bakker for his support, guidance, and friendship during my work at VCU UAV Laboratory. As well, I would like to thank my fellow graduate students, Matt Leccadito, Tom Carnes, and Garrett Ward for their aid and friendship. Last and most importantly, I would like to thank my family for their love and support throughout my life.

Table of Contents

Abstract

The purpose of this research was to develop a mobile ad-hoc network for collaborative Unmanned Aerial Vehicles (UAVs) based on a mesh networking standard called IEEE 802.11s. A low-cost, small form-factor, IEEE 802.11a based wireless modem was selected and integrated with the existing flight control system developed at Virginia Commonwealth University (VCU) UAV Laboratory. A self-configurable user-space application on the wireless modem was developed to provide functionality to collaborative algorithms, and to monitor the performance of the wireless network. The RAMS simulator, developed at VCU, was upgraded to support the simulation of advanced networking capabilities by integrating with a simulator called ns-3. The reconfigurability and performance of the IEEE 802.11s mesh network was validated and evaluated by conducting real-world flights. The results show that the IEEE 802.11s is a promising solution for collaborative UAV applications.

List of Tables

List of Figures

List of Abbreviations

AODV	Ad-hoc On-demand Distance Vector
CSMA	Carrier Sense Multiple Access
CUS	Collaborative UAV System
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
FCS	Flight Control System
GCS	Ground Control Station
GPS	Global Positioning System
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network
MCS	Mission Control System
OLSR	Optimized Link State Routing
PERR	Path Error
PLP	Packet Loss Percentage
PLR	Packet Loss Ratio
POSIX	Portable Operating System Interface
PREP	Path Reply
PREQ	Path Request
RANN	Root Announcement
RSSI	Received Signal Strength Indicator

UAV	Unmanned Aerial Vehicle
UDP	User Datagram protocol
UDS	Unix Domain Socket
VACS	VCU Aerial Communication Standard
VCU	Virginia Commonwealth University
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Connection

Chapter 1: Introduction

1.1 Overview

Unmanned systems have emerged as promising machines that can execute missions that are otherwise difficult or impossible for humans to accomplish. Unmanned Aerial Vehicles (UAVs), a subset of unmanned systems, have gained acceptance into a wide range of military and civilian applications as mentioned in [1]. Some of the popular military applications include surveillance, reconnaissance, and meteorology missions. Law enforcement, border patrol, search and rescue, wildfire detection and communication relay are some of the notable civilian applications. UAVs are operated by trained personnel because of the complexities involved in maneuvering them. In order to alleviate this problem, the UAVs are usually equipped with an autopilot for self-guided navigation and control.

Due to the recent advancements in computer and wireless technology, multiple autonomous UAVs operated by a single operator, called Collaborative UAV Systems (CUS), have become realizable. Teams of collaborative UAVs can solve complicated problems such as surveillance in a large area, by autonomously dividing up the area to be searched into smaller sub-areas and searching them individually. This could be solved more efficiently with a computer dynamically rather than an operator assigning tasks statically all at once. The functioning of a CUS involves a high-level of interaction between autonomous UAVs. Furthermore, the UAVs must always be available to the operator who assigns tasks and sends control commands [2]. Issues related to control and communications between these vehicles are discussed in [3].

The most critical requirement for the functioning of a CUS is the existence of a robust and reconfigurable wireless network. Developing such a network for collaborative UAVs is a

challenging task because the network must adapt to the dynamically changing link conditions over time. The wireless link quality is mainly dependent on the distance of separation between nodes and the orientation of the radiation pattern of these nodes, which are highly dynamic for a typical UAV mobility model. An increase in the distance of separation attenuates the transmitted signal because of radio wave propagation losses. The orientation of the aircraft dictates the orientation of the antenna mounted on it, which in turn dictates the radiation patterns around the UAV. The network has to reconfigure itself in order to establish a reliable and robust communication between any two nodes directly or indirectly. That is, the network must be reconfigurable in nature and support communication over multiple hops. The time after which the network should reconfigure itself is highly uncertain and is dependent on the relative mobility and link variability between the nodes. All of these factors contribute to the complexity involved in building a wireless network for CUS.

Wireless networks can broadly be classified based on infrastructure and mobility. If the communication between nodes in a network relies on a central infrastructure for network management then the network is classified under infrastructure based networks. For example WLAN or IEEE 802.11 networks have access points managing the infrastructure and communication between stations in the network. Otherwise, the network is classified under ad-hoc networks. For example nodes in an IEEE 802.11s mesh network are capable of functioning both as an access point and a station [4]. If the nodes in the network are geographically mobile then the network is classified under mobile wireless networks. Otherwise, the network is classified under static wireless networks. Mobile ad-hoc network (MANET) is a special category of wireless network that has both the characteristics of a mobile and ad-hoc wireless network. A network of collaborative UAVs can be categorized as MANET because of the mobility

associated with the nodes in the network and the communication requirements noted above. MANETs widely apply routing protocols such as HWMP (IEEE 802.11s), BATMAN Adv, OLSR, AODV, and DSR. A review of the routing protocols for MANETs is given by [5, 6].

This research presents the development of a mobile ad-hoc network for collaborative UAVs based on the IEEE 802.11s standard. A low-cost, small form-factor, IEEE 802.11a based wireless modem was selected and integrated with the existing Flight Control System (FCS) developed at Virginia Commonwealth University (VCU) UAV Laboratory. The wireless modem was installed with an embedded Linux distribution called OpenWrt and an open-source reference implementation of IEEE 802.11s called open80211s. The stability issues that arose due to the high mobility of UAVs were fixed in the open80211s package. A self-configurable user-space application for the wireless modem was developed to provide support and functionality for the collaborative algorithms. A collaborative UAV simulator called RAMS developed at VCU was integrated with the widely used network simulator called ns-3. A framework for launching the application programs during the simulation run-time was developed in the simulator. The reconfigurability and performance of the IEEE 802.11s mesh network was validated and evaluated by conducting real-world flights.

1.2 Problem Statement

In order to facilitate the interaction between the collaborative UAVs, a wireless modem must be selected with the following minimum requirements. The bandwidth of the modem should be at least 1 Mbps. The modem must be low cost and commercially available so that it can be deployed in large numbers. The size of the wireless modem must be small so that it can fit in a small UAV such as a hand launchable glider platform, and the modem should provide support

for an Ethernet and a serial interface so that it can be easily integrated with already existing Flight Control Systems via standard communications connections.

The wireless modem must be installed with an embedded Linux distribution which has support for most ad-hoc networking protocols and wireless networking tools. The support for more ad-hoc networking protocols gives the flexibility to experiment as well as evaluate the protocols under the same platform. Wireless networking tools are essential to configure and evaluate the network. The most widely used open-source reference implementation of IEEE 802.11s for ad-hoc networking applications is open80211s [7], which has not yet been extensively tested in highly dynamic environments. Hence, the functioning of the protocol and the implementation has to be validated in a collaborative UAV scenario. Any stability issues that may arise in mobile scenarios must be fixed before the final deployment. Stability is important in mission critical applications to prevent system crashes and possible damage to expensive equipment.

The networking software architecture of the CUS should be such that the FCS software remains compatible across different modems. An application that can run in the user-space of the wireless modem has to be developed which can provide required functionality to collaborative algorithms while abstracting away the network layer addressing of nodes from the FCS. When using the HWMP protocol the existence of all the hosts (FCS module or GCS computer) need to be globally known, only the paths to the nodes will be discovered by the protocol. The user-space application needs to dynamically discover and maintain the list of the UAVs in the network and provide the information in the form of UAV identity numbers. Furthermore, the application must be able to collect the wireless network statistics and send them over to Ground Control Station (GCS) in regular intervals in order to analyze the performance of the network.

The network module of RAMS simulator [8] developed at VCU supports low-fidelity simulation of CUS. It includes a rudimentary communications link simulation capability. This simulation capability currently supports the simulation of packet drops based on distance of separation between UAVs and collisions. An accurate model of the wireless network is necessary to study the applicability of a routing protocol or a wireless technology to an application. A model of the Wi-Fi modem used in the real-world test-bed must be incorporated in the RAMS simulator to fully evaluate the performance of the wireless network under different scenarios. Rather than modeling the routing protocols and standards from scratch, a network simulator called ns-3 [9] must be integrated with the RAMS simulator. Furthermore, optimum values for the configurable parameters of the modem and the routing protocol can be achieved by performing simulations.

A framework that can execute the application programs that are designed for Linux operating system would be of a great value to the RAMS simulator. Such a framework would save time and effort spent in modeling the functionality of the applications in the simulator. Furthermore, the simulator must be able to load the application programs during run-time, so that frequent compilations of simulator code during the development phase would not involve the application code as well. That is, the compilation of the userspace communications application must be separated from the simulator.

The evaluation of mobile ad-hoc networks usually involves performing hundreds of experiments with varying parameters of interest – the most notable of which is the varying distance between nodes. The l RAMS simulator includes a user-driven mobility model that requires that the waypoints need to be assigned to multiple planes, each plane separately, on

every single experiment. Thus, the user-driven mobility model must be extended with random waypoint models to accelerate the experimentation process for this work using RAMS.

1.3 Document Layout

The remainder of this document is laid out as follows. Chapter 2 gives a background on the IEEE 802.11s, and VCU collaborative UAV test-bed and simulator. The chapter also surveys the related work that has been done in developing and evaluating MANETs. Chapter 3 describes the specifications of the selected modem and the factors that were used in selecting the wireless modem and the underlying routing protocol for this application. Furthermore, it also describes the architecture of the mesh node and user-space application, and presents and describes a block diagram of the collaborative UAV system. Chapter 4 discusses the simulation framework that was developed for simulation of application programs and the architecture of ns-3 integrated with the RAMS simulator. Chapter 5 presents the results obtained from real-word flights and simulations. Finally, Chapter 6 concludes the paper by pointing out challenges and scope for future research.

Chapter 2: Background

This chapter provides a background on IEEE 802.11s, RAMS and ns-3 simulators, and related work. Section 2.1 gives a brief overview of the IEEE 802.11s standard and the operation of Hybrid Wireless Mesh Protocol (HWMP). Section 2.2 discusses the architecture and gives a brief overview of the RAMS simulator. Section 2.3 discusses the architecture and features of ns-3 simulator. Section 2.4 gives a brief overview of the collaborative UAV system built in the UAV Lab at VCU. Section 2.5 covers the prior work that was being done in the development and evaluation of mobile and ad-hoc networks.

2.1 IEEE 802.11s

The IEEE 802.11s mesh networking standard provides a framework for the implementation of path selection protocols and metrics. The path selection protocol uses metric information to establish paths between nodes in a network. IEEE 802.11s defines a default path selection protocol called Hybrid Wireless Mesh Protocol (HWMP) and a metric called Airtime Link Metric (ALM). Although, the standard addresses a lot of terms and issues, only those which are relevant to the context of MANET are described in this section.

The frames in IEEE 802.11 standard are classified into data, control and management; data frames are used to encapsulate data pertaining to higher layers; control frames are used for acknowledgements and reservations; management frames are used to setup, organize and maintain WLAN. The 802.11s standard adds a mesh control field to the current 802.11 frame to provide multi-hop capabilities at the MAC layer. Fig 2.1 shows the frame format of the IEEE 802.11s.

The IEEE 802.11s standard defines four control frame formats abbreviated as PREQ (Path Request), PREP (Path Reply), PERR (Path Error), and RANN (Root Announcement). Each node in the network holds a sequence number which is normally modified by the same node, but in special cases another node in the network may modify as well. Sequence numbers are introduced to distinguish between the new and old information in the forwarding tables. Every node stores the routing information as a forwarding table. Each row in a forwarding table contains the destination address, next hop address, and sequence number.

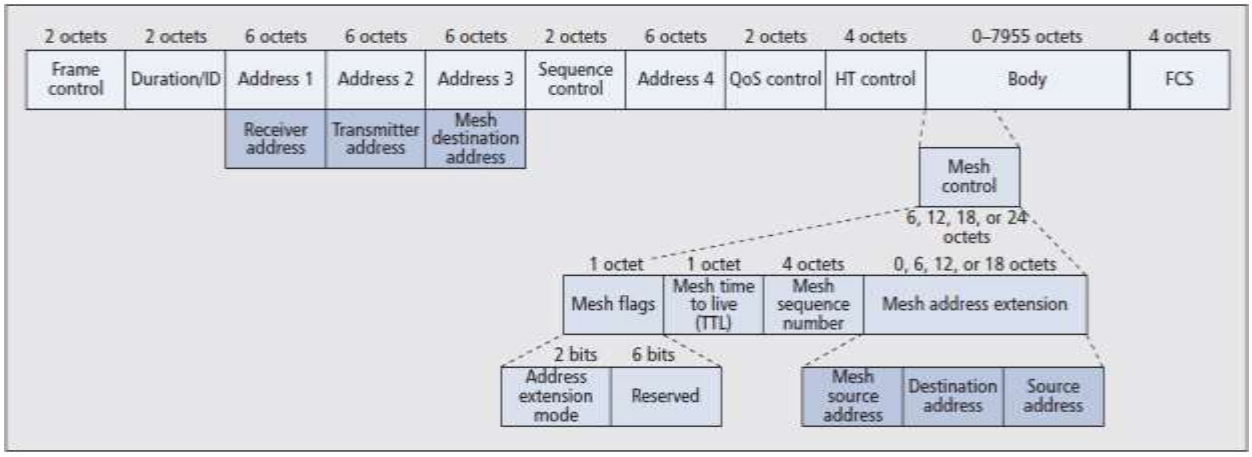

Fig 2.1. IEEE 802.11s Frame Format [10]

A mesh node in a network has the following functionalities. Mesh discovery, peering management, beaconing and synchronization, mesh coordination function, mesh path selection and forwarding, and intra mesh congestion control. Mesh discovery is performed by periodically sending beacon frames to the neighboring nodes in the network. Neighboring nodes means the nodes which share the same wireless medium or channel. The discovered neighboring nodes which meet the requirements to communicate will be made peers to the given node after exchange of control messages as per the peering protocol. For a proper exchange of frames without collisions the mesh nodes are synchronized by using the synchronization and

coordination procedures. Once the nodes are discovered and peers are established, a node can then explore paths to different nodes in the network by using a path selection protocol like HWMP.

HWMP supports two modes of operation, on-demand and proactive. In the on-demand mode any mesh node can request a route to another node in the network. Alternatively in the proactive mode, all mesh nodes always maintain a path to the root node and any traffic arising at a node is first forwarded to the root node and then to the destination. The proactive mode establishes communication between source and destination instantly on a request because a path from source to destination is always available through the root. The on-demand mode however first needs to create a path to the destination, before the communication begins, causing a delay. The proactive mode is prone to congestion at the root because all the traffic has to be routed through the root, whereas on-demand mode saves network bandwidth by creating paths only upon request.

In the on-demand mode if a source node S has to send data to a destination node D, then it first has to discover a path from S to D. The process is initiated by S by broadcasting a PREQ frame to all its neighboring nodes. If none of the neighboring nodes has a path to D then the PREQ frames are further broadcasted by each of the nodes. Finally, when the PREQ frame reaches D, the node D sends back a unicast PREP frame to S. During the travel of PREQ from S to D, a reverse path from each intermediate node to S is established. While a forward path is established from each intermediate node to D during the travel of PREP from D to S. S creates an entry for D in its forwarding table on reception of PREP from D or an intermediate node in special case.

The proactive mode requires one of the nodes to be configured as a root mesh node. The root node periodically broadcasts proactive PREQs which are received by all the connected mesh nodes in the network. Two types of network discovery with proactive PREQ mechanisms exist, one which mandates a PREP from the mesh node back to root and other without. In the former mode, the root node has paths to all the mesh nodes in the network and vice-versa, whereas in the later mode only mesh nodes have paths to the root. The routing tables are updated when the PREQs are disseminated from the root node to the mesh nodes. The root node updates its routing table on receiving PREPs from the mesh nodes.

The HWMP protocol defines a timeout period for each established path. A path created in the on-demand mode is invalidated after a preset timeout period. The purpose of the timeout parameter is to invalidate the old paths and explore better paths. The timeout parameter can be set to any value based on the requirements of the application and is set by default to 5.12 seconds in the open80211s implementation.

By default the Airtime Link Metric (ALM) is calculated by the equation:

$$Ca = \left[O + \frac{Bt}{r}\right] * \frac{1}{[1 - ef]}$$

Where O is the channel access overhead, Bt is the length of the test frame in bits, r is the bit rate in Mbps, and ef is the frame error rate. Frame error rate is the probability that the frame of size Bt transmitted at a rate r gets lost due to corruption of bits. In the open80211s implementation, O is always assumed to be 1 and value of Bt is 8192. Values of r and ef are variable and are obtained from the physical layer. The IEEE 802.11a physical layer (PHY) can transmit bits at a rate of 6, 12, 18, 24, 36, and 54 Mbps. The ef value ranges between 0 and 1.

Since, ALM is directly proportional to the frame error rate; a higher metric value means that the quality of the links in a path is low.

2.2 RAMS

A low-fidelity discrete-event multiple agent simulator called RAMS was designed and built by the graduate students in the UAV Laboratory at VCU. The goal of the project was to build a low-fidelity simulator which can simulate high level decision algorithms executed by a team of collaborative autonomous UAVs. A medium-fidelity model of the wireless network connecting the UAVs was incorporated in the simulator. A mobility model for an UAV was included in the simulator based on simple flight dynamics. The simulator supports two modes of operation, namely, real-time and speedup. The simulation time in real-time mode advances with the real-time whereas in speedup mode it advances based on a desired speedup factor. The speedup parameter can be set dynamically during the run-time of the simulation.

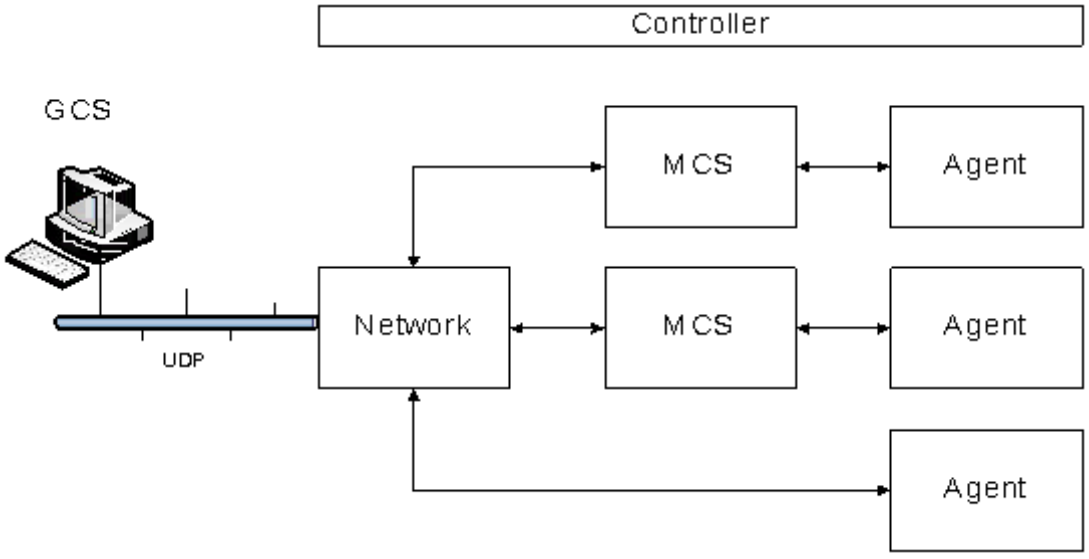

Fig 2.2. Architecture of RAMS Simulator [8]

The architecture of the simulator is shown in Fig 2.2. The simulator was designed to be modular and was implemented in C++ to leverage the objected oriented functionality of the language. The modular design provides the flexibility to tie different modules together based on the application. Each module in the simulator is a Portable Operating System Interface (POSIX) thread which gets executed periodically. The inter-module communication is established by standard C++ queues with concurrency support. The functionality of the modules of RAMS simulator will be briefly described in the following paragraphs. A more detailed description of the RAMS simulator is available in [8].

The Agent module models the autonomous vehicle, in this case, a UAV. The agent is modeled as a point mass in 3D space. The point's location in Longitude Latitude Altitude (LLA) coordinate system is always defined. The agent module simulates the motion and attitude of the UAV during banking, accelerating and descending. The agent module acts based on the waypoints received from Ground Control Station (GCS) or the Mission Control System (MCS) module. During single vehicle operations, the operator sends waypoint to the Agent via the GCS. During collaborative operations the MCS module sends waypoints to the agent module based on the individual vehicle's task assignment. The MCS module executes the algorithms required for collaborative missions.

The network module simulates the packet drops and collisions which naturally occur in a real-world wireless communication system. The wireless channel is modeled by simulating collisions, that is, by dropping packets transmitted by two different nodes with overlapping transmission times. The network module uses the Friis propagation loss model to model the propagation path losses that occur before the transmitted signal reaches the receiver. The packets

with the received signal strength below a certain threshold are dropped by network module. The communication with GCS is established over Ethernet applying user datagram protocol (UDP).

The controller module regulates the speed of the simulation, instantiates modules, and provides scheduling information to the modules in every iteration. The necessary and some mandatory modules for simulation are created by the controller at the start of the simulation. The modules in the simulation can fetch the simulation time from the controller. Based on the frequency at which each module was set to execute, the controller computes the time instant at which the module's thread must start execution.

2.3 Ns-3

A survey of network simulators available in academia and industry is given in [11, 12]. In this project, a discrete-event network simulator called ns-3 was chosen due to its modular architecture and the support for simulation of wide range of ad-hoc networking protocols. Ns-3 is a discrete-event network simulator targeting a research and educational audience. The ns-3 simulator models five layers of the International Standardization for Organization (ISO) Open Systems Interconnection (OSI) model, namely, application, transport, network, link, and physical. Ns-3 supports simulation in real-time mode, where in, the events in simulation are executed at the exact time as they would happen in real-world.

The ns-3 simulator's configuration framework provides an easy way to configure the simulation parameters from an experimental setup script. A script that instantiates the necessary objects of simulation is known as an experimental setup script and typically all the simulation parameters are set in this script by obtaining references to internal objects. Ns-3, however,

provides a configuration interface which can be used to set the parameters of simulation by just specifying the path to the simulation objects.

The user applications that run on a node on ns-3 extend an abstract base class called Application. The Application class provides function prototypes to specify the start and stop times for an application in terms of simulation time. Like in a Unix/Linux based operating system, the ns-3 provides socket programming interfaces which let you send and receive packets over the network. The packets can be unicasted and broadcasted in a similar way as being done on a Unix/Linux based operating system. The internet protocol layer takes care of address assignment and routing. Several ad-hoc routing protocols are supported like OLSR, AODV, DSR, and DSDV. Several network devices are implemented which are based on Wi-Fi, CSMA, WiMAX, LTE, loop back, mesh point, and point-to-point technologies.

The architecture and implementation of the IEEE 802.11s standard for ns-3 is well described in [13, 14].

2.4 VCU Collaborative UAV System

The collaborative UAV system was developed in the UAV Laboratory at VCU as part of the master's thesis of a fellow graduate student [15]. In his project he added a co-processor to the flight control system to execute the collaborative algorithms. Co-processor is based on ARM architecture and runs at 600 MHz clock frequency. The Gumstix platform was used for the co-processor which is capable of communication over Ethernet. A collaborative search algorithm was implemented and tested in simulations a well as real-world flights [15].

The miniFCS described in [16] was developed by another graduate student in the UAV Lab which has a small form factor suitable for small UAVs, as seen in Fig 2.3. The glider platform and the placement of the miniFCS is shown in Fig 2.3.

Fig 2.3. Glider UAV platform, placement of miniFCS [16]

The flow of information and the modules that were used in the collaborative UAV system developed by [15] are described as follows. The telemetry data was sent periodically from the FCS to the MCS module. The telemetry information received by the MCS module was sent to the GCS via the XBee-PRO 900 RF wireless module. The communication between FCS and MCS was serial, and MCS and the wireless modem was also serial. Fig 2.4 shows the system architecture of the collaborative UAV system.

The wireless modem was operated in API mode, where in, the destination address of the packet can be specified while sending a packet. Each wireless modem was assigned a 64-bit address. A severe limitation of the wireless modem was that the maximum size of the data packet supported was only 72 bytes. In order to overcome this limitation any data packet that was larger than 72 bytes was fragmented to smaller frames and sent to the destination with a frame ID. The frames at the receiver were recombined based on the ID and then data packets are supplied to the MCS module. The frame aggregation and fragmentation increased the protocol overhead and

thereby decreased the maximum number of data bytes sent per second. This drawback along with lower data-rate of the modem limited the performance and reliability of the communication system.

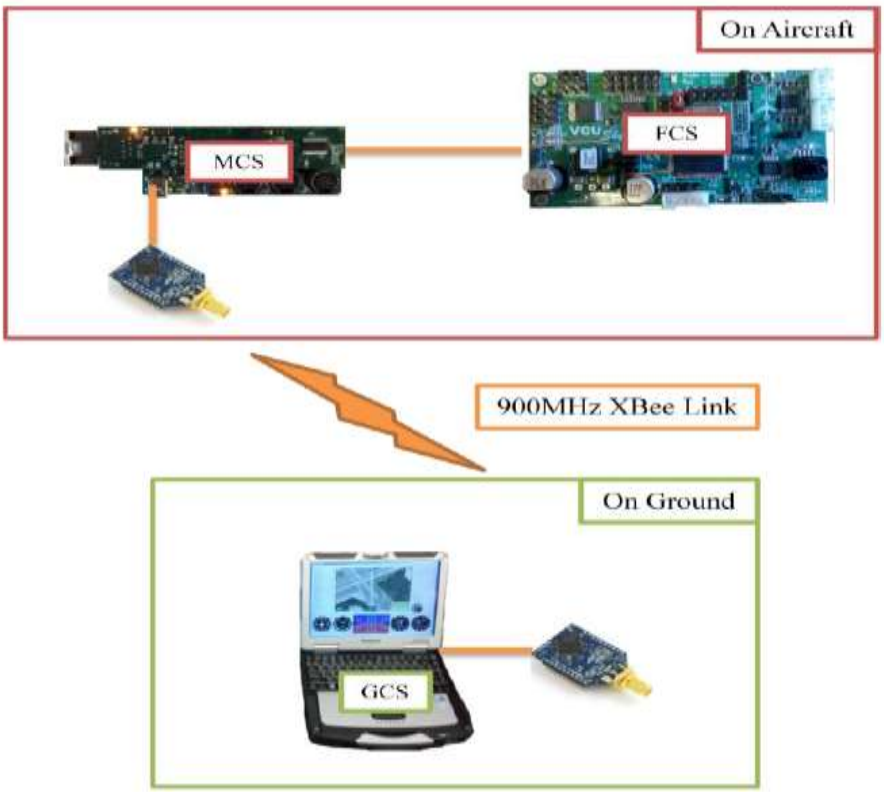

Fig 2.4. System Architecture [15]

The communication between the FCS and GCS abides the VCU aerial communication standard called VACS. The VACS packet format is shown in Table 2.1. The SYNC bytes mark the start of a data packet. Destination and Source are the destination and source identification (ID) numbers of the planes. The ID of the GCS is zero and the broadcast ID is 255. Message high and low together forms the message ID of the data packet indicating the type of the message. Length-high and length-low together form a number indicating the size of a packet.

The data field contains the payload with a maximum possible size of 1024 bytes. The correctness of the packet is checked by using the checksum which lies in the last two bytes of the packet.

Table 2.1: VACS Packet Format

Sync1 (0x76)	Sync2 (0x63)	Destination	Source	Mess. High	Mess. Low	Length High	Length Low	Data	Check High	Check Low

2.5 Related Work

This section discusses the prior work that has been done in building ad-hoc networks with multiple nodes, the techniques used in the performance evaluation of such networks, and the implementation issues that arise in building them.

An ad-hoc mesh network of mobile and stationary nodes was constructed at the University of Colorado at Boulder [17]. The goals of their project were to build a wireless ad-hoc network and embed a monitoring architecture inside each radio for detailed performance characterization and analysis of the network. The test-bed consisted of ground nodes as well as aerial nodes (UAVs) as shown in Fig 2.5. A network node was built with a processor board, 2.4 GHz IEEE 802.11b wireless card, a 1W bidirectional amplifier and a GPS device. Dynamic source routing (DSR) protocol was implemented on a Linux platform. Experiments were performed in two scenarios. In first, the UAV acts as a radio node that connects disconnected ground radios. In second, two UAVs communicate with each other to extend a third UAVs' operational range. The same hardware and software are used for both UAV and ground nodes for making the network statistics easy to analyze. Experiments were performed to test throughput, connectivity, congestion, user experience, node failure, and range. A tool called *netperf* was used

to measure the throughput between pairs of nodes for 5 sec periods. The connectivity of network was tested by having each node pinging a random destination node every second. Congestion was tested by measuring delays and throughputs when there are competing data streams. Subjective tests were performed by assessing the performance of network applications as perceived by a user browsing the internet. The ability of the network to recover from node failures was tested by making a node continuously go on and off for every 60 sec. Range tests were performed by measuring throughput versus distance of separation between nodes. This work differs from their work in the protocol used for the MANET and the technique used to evaluate the network.

Fig 2.5. UAV nodes (left), ground nodes (right) [17]

In [18] experiments were conducted with above mentioned hardware and software to prove the feasibility of mesh networking architecture for UAVs. Experimental results showed that operational range of the network increased significantly by applying a mesh networking architecture. The authors mention that the downsides of the mesh architecture are an increase in latency and a decrease in throughput of the network. Fig 2.6 shows the architecture of the unmanned aircraft system developed at University of Colorado at Boulder.

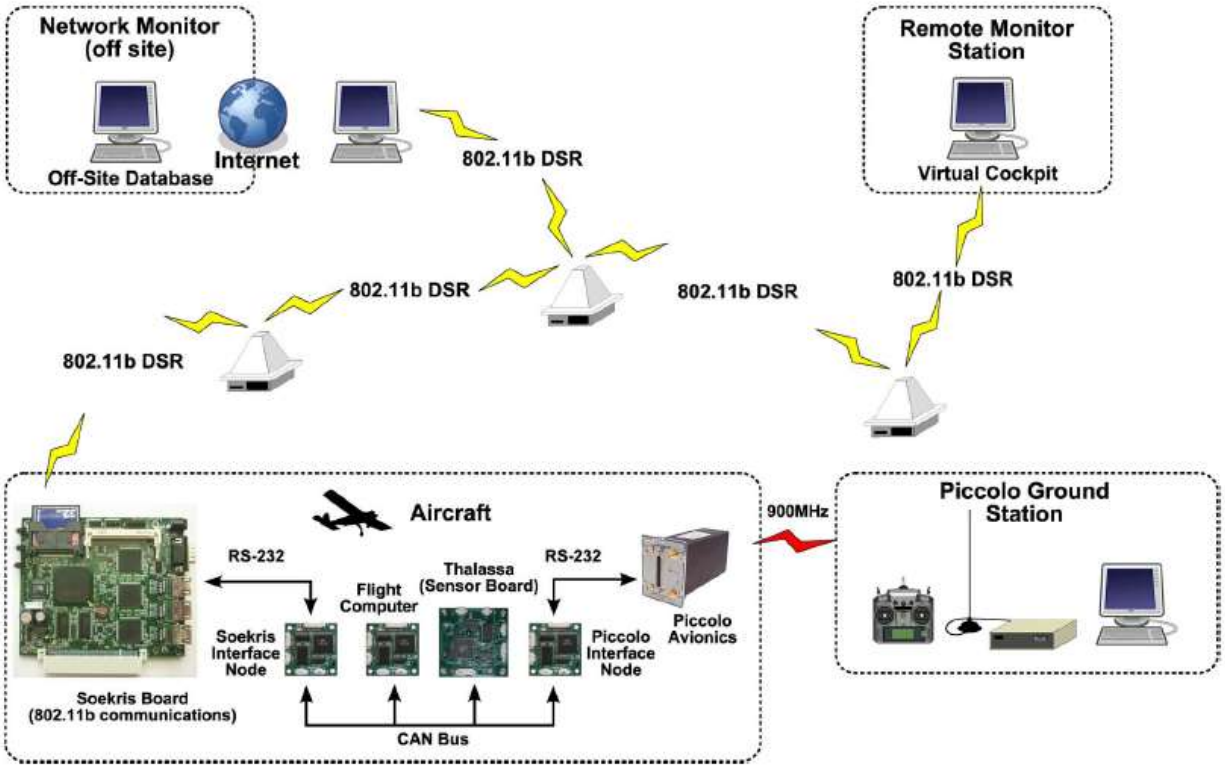

Fig 2.6. Architecture of UAS developed at University of Colorado at Boulder [18]

A wireless mesh network based on quadrocopters was built at the University of Bern, Switzerland [19]. The end systems were two stationary notebook computers and the UAVs were used as airborne relays. The wireless modem was based on IEEE 802.11s and open-mesh platform. The modem used had IEEE 802.11 b/g wireless and UART host interfaces. The mesh network was evaluated in single-hop and multi-hop airborne relay scenarios. The software for automatic establishment of connections between end systems (notebooks) was developed. Fig 2.7 shows the UAV platform and wireless modem used, and Fig 2.8 shows the software architecture of the application running on modem. The application uses the library libuavext to handle the communication between the UAVs. Libuavint is used to communicate with the flight control system over the serial line. The application developed handles two scenarios namely: a single UAV acting as an airborne relay and a scenario with two UAVs acting as airborne relays.

The end-to-end UDP and TCP throughputs of the built wireless mesh network were measured and reported. This research is similar to their work with regard to establishing communication between end systems. However, the reconfigurability of the IEEE 802.11s mesh network was not evaluated in their work.

Fig 2.7: UAV and radio used at University of Bern, Swtizerland [19]

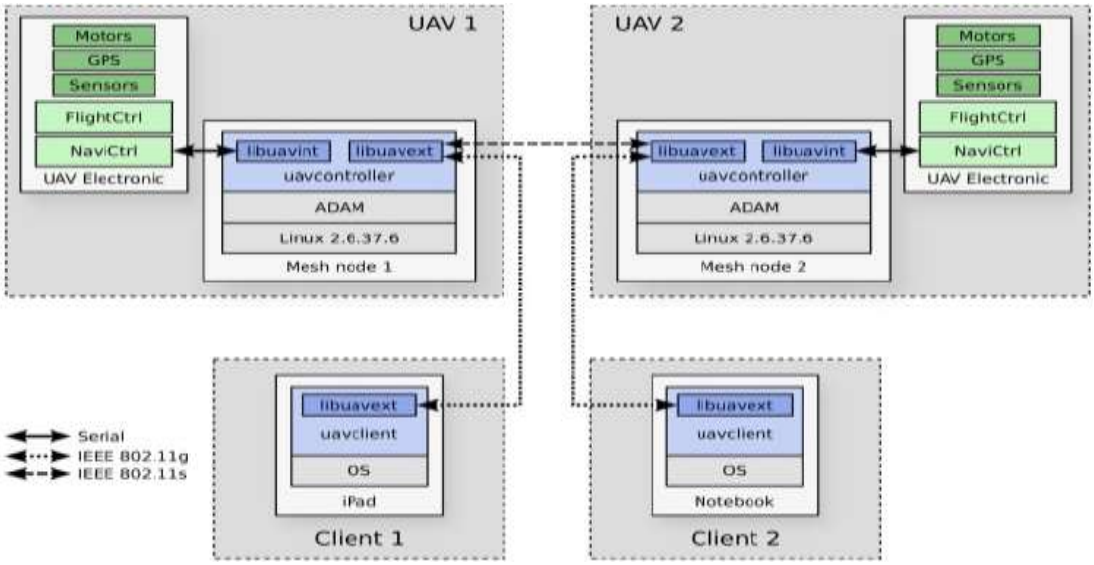

Fig 2.8. Architecture of UAS developed at University of Bern, Switzerland [19]

In [20] the authors evaluate the mesh network by measuring throughput with varying number of hops. Furthermore, the effect of the routing protocol overhead on the throughput performance was studied. The mesh protocol was implemented as a kernel module. Their paper mainly focuses on the mesh networking, and the interworking with other IEEE 802.X networks

but not the MANET aspect. An open source Linux based operating system distribution called OpenWrt Kamikaze 8.09 for wireless devices was used by them [21]. The software stacks used in the Linux were mac80211, open80211s, ath5k, and net80211 [22]. The software platform used for their project and this project are almost the same. However, the authors only evaluate the mesh network with static nodes.

Building and installation of open80211s packages was discussed in [23]. The installation was carried out in ubuntu operating system. The objective of their work was actually to build a test-bed based on IEEE 802.11s and carryout throughput measurements under different network scenarios. So, they selected the wireless card which is compatible with the mac80211 package, and the driver with highest compatibility. The P54 driver was used on Linux kernel 2.6.29. D-Link USB adapter was the wireless radio used to test the packages. A package for 802.11s was also added to the ns-3 simulator and the results obtained were compared to real world test-bed consisting of laptops as network nodes. This research differs from their work in the environment the protocol was tested, mobility of the network nodes, and the platform used to build the test-bed.

A mobile wireless network of Micro-Air vehicles called SensorFlock was developed based on IEEE 802.15.4 technology radios in [24]. The UAV platform and the radio used are shown in Fig 2.9. Experiments were conducted to characterize the plane-to-plane and, plane-to-ground link in one, two, and five plane scenarios. The samples of GPS location, attitude information, and RSSI were sent to GCS periodically. The results showed that the RSSI varied as a function of distance and roll angle. The authors concluded that the variation in RSSI was prominent in plane-to plane rather than plane-to-ground channel.

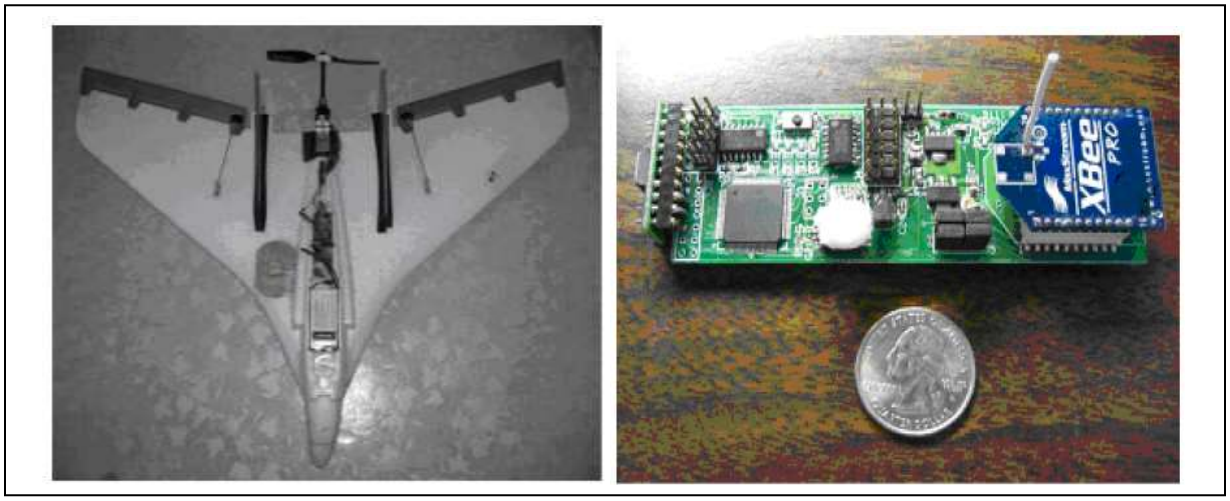

Fig 2.9. The UAV platform and the IEEE 802.15.4 technology radio used in [24]

In [25] the experimental and simulation study of ALM and HWMP was carried out. The authors show the variation of ALM over time in a static wireless mesh network of four nodes. By fixing some flaws in the protocols the authors managed to increase the performance significantly.

In [26] the issues related to link measurement experiments in the field were discussed. The experiments were conducted with COTS-based 2.4 & 5 GHz IEEE 802.11 modems as a communication data link between a UAV and a ground node. A plot of RSSI as a function of distance was shown in their results.

In [27] the performance evaluation of an 802.11-based MANET was carried out for a network of eleven static ground nodes, one moving ground node and one UAV node. The ad-hoc networking protocol used was an extended Open Shortest Path First (OSPF) implementation. The authors analyzed the variation of link-up times of several nodes in the network and proposed

heuristics to alleviate the link stability problems. The metric calculation was modified to result in a better value for stable paths.

The novelty of this research lies in the design of a networking software architecture for a CUS, and the approach used for the validation and evaluation of a MANET.

Chapter 3: Test-bed Development

3.1 Hardware Platform

A 5 GHz wireless modem was selected as a communications data link between UAVs because not only the spectrum is less crowded compared to 2.4 GHz, but also the higher frequency leads to more bandwidth. A higher carrier frequency signal suffers from more attenuation and path losses. However, the range can be extended by utilizing multi-hop ad-hoc networking techniques. Based on the availability and support for open-source software, the Ubiquiti Bullet5 was selected as the modem for our collaborative platform. It has an IEEE 802.11a PHY, 16 MB RAM, 4 MB flash memory, a 100 MHz Atheros 2313 processor, and a serial and Ethernet interface [28]. The N-type antenna connector on the modem was replaced by an RP-SMA connector in order to reduce the size and weight of the modem as shown in Fig 3.1. The UAV platform used for this research is shown in Fig 3.2. The modem can be located in Fig 3.3, sitting below the wing on the right side of the fuselage. The Single Board Computer (SBC) on the left side of the fuselage is the mission control system for running collaborative algorithms which is part of the FCS.

Fig 3.1. The modified bullet5 modem having the N-type connect replaced by RP-SMA

Fig 3.2. Glider UAV platform

An omnidirectional 3 dBi dipole antenna [29] was installed for the GCS as well as on the UAVs. An omni-directional antenna with low gain has a higher beam width of the main lobe in elevation plane. A high beam width in the elevation plane is essential in order to maintain robust UAV-to-UAV and ground-to-UAV communications. The omni-directional antenna for the GCS was mounted on a 10 ft. pole in the field as shown in Fig 3.4.

Fig 3.3. Physical location of wireless modem (right) inside UAV

Fig 3.4. The field antenna and GCS node mounted on 10 ft. pole

3.2 Software Platform

The wireless modem was installed with the OpenWrt embedded Linux distribution [21]. After experimenting with different versions of kamikaze and bleeding-edge branches of OpenWrt, the version used was attitude adjustment 12.09 r35531. The attitude adjustment was more stable than other versions of OpenWrt, although needed some small bug fixes. The software architecture of the IEEE 802.11s mesh node, userspace applications, kernel modules, and drivers used, and the network interfaces created is shown in Fig 3.5.

Some of the user-space tools used include *iw*, *iwconfig*, *iperf* and *mii-tool*. *iw* and *iwconfig* are applications for configuring the wireless interface, *mii-tool* is used to configure the Ethernet interface, and *iperf* can be used to measure the network performance. The libraries

namely, pthread and GNU C, are installed on the node so that the multithreaded userspace application developed can run on the mesh node.

The Linux kernel modules namely mac80211, cfg80211, ath5k, and compat must be included in the building process in order to successfully create a Linux image for an IEEE 802.11s mesh node using OpenWrt. Mac80211 kernel module is a framework to write drivers for the Wi-Fi devices. Cfg80211 kernel module provides the 802.11 configuration API. The packages were compiled into the kernel as kernel modules to speed up the debugging process because recompiling the kernel takes a relatively long time compared to compiling the kernel modules alone.

A network interface called *wlan0* was created to receive packets from the wireless device and an interface called *eth0* to receive packets from the Ethernet. In order to send and receive packets over the two interfaces, they need to be assigned a different IP address. Handling two IP addresses for a single modem is unnecessary, hence the two network interfaces are bridged together to form a third interface called *br-lan*. The *Brctl* tool was used to bridge the wireless and Ethernet interfaces. The *br-lan* interface is assigned an IP address statically and all the incoming traffic to this interface is forwarded to both the wireless and Ethernet interfaces. The disadvantage of this approach is that, the higher layers can't distinguish between the traffic coming from Ethernet or the wireless interface. However, this could be overcome by assigning different port numbers to UDP sockets on the transport layer.

The following wireless configuration worked best in the experiments conducted with a single UAV. The link was found to be more robust when the transmit power was 17 dBm. The lowest available channel number, 36, is using the lowest carrier frequency and, therefore, is more

robust to signal attenuation. The wireless configuration of the mesh node built and deployed is listed in Table 3.1.

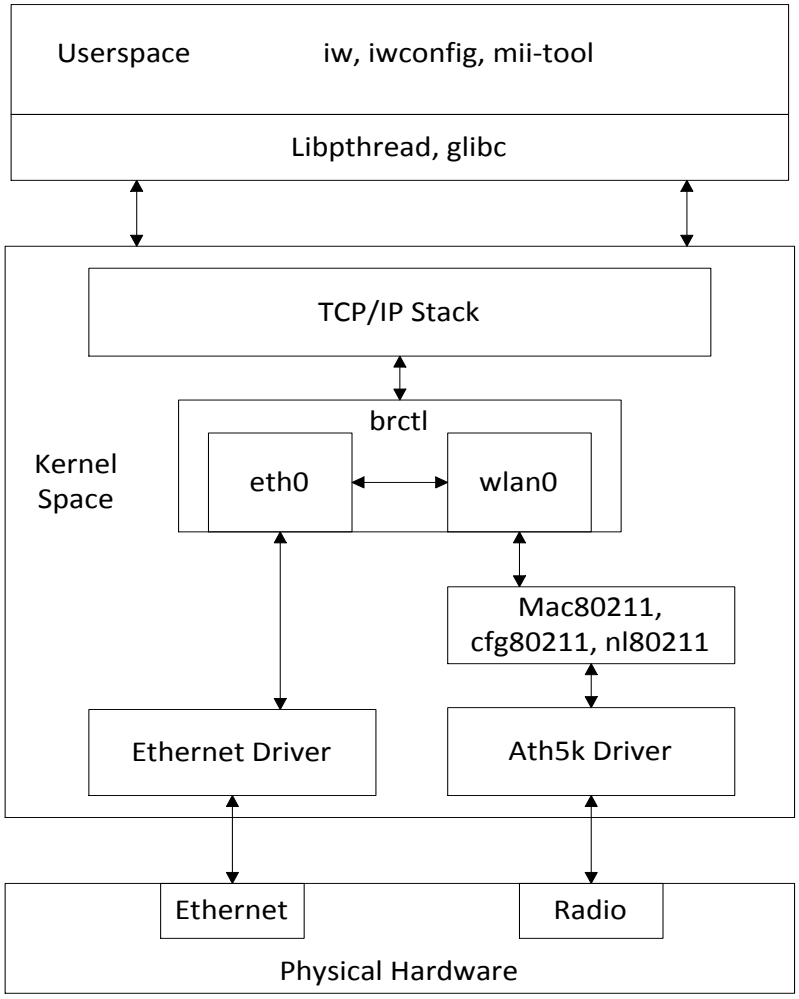

Fig 3.5. IEEE 802.11s mesh node software architecture

Table 3.1. OpenWrt Wireless Configuration

Radio Options	Values
hwmode	11a
type	Mac80211
txpower	17
channel	36
network	lan
mode	mesh

3.3 Userspace Application

The CUS network architecture is shown in Fig 3.6. The wireless modems installed on the UAVs and for the GCS are all connected to each other by an IEEE 802.11s mesh network. The user-space application handles messages that are defined in the Virginia Commonwealth University Aerial Communications Standard (VACS). This standard defines a header for every message, which includes a source, a destination, and a message identity number. The source and destination identity are unique numbers where the GCS is allocated the number 0, UAVs are identified by the tail number, and 255 is used for broadcasting. The user-space application provides support and functionality to the collaborative algorithms by providing the following services.

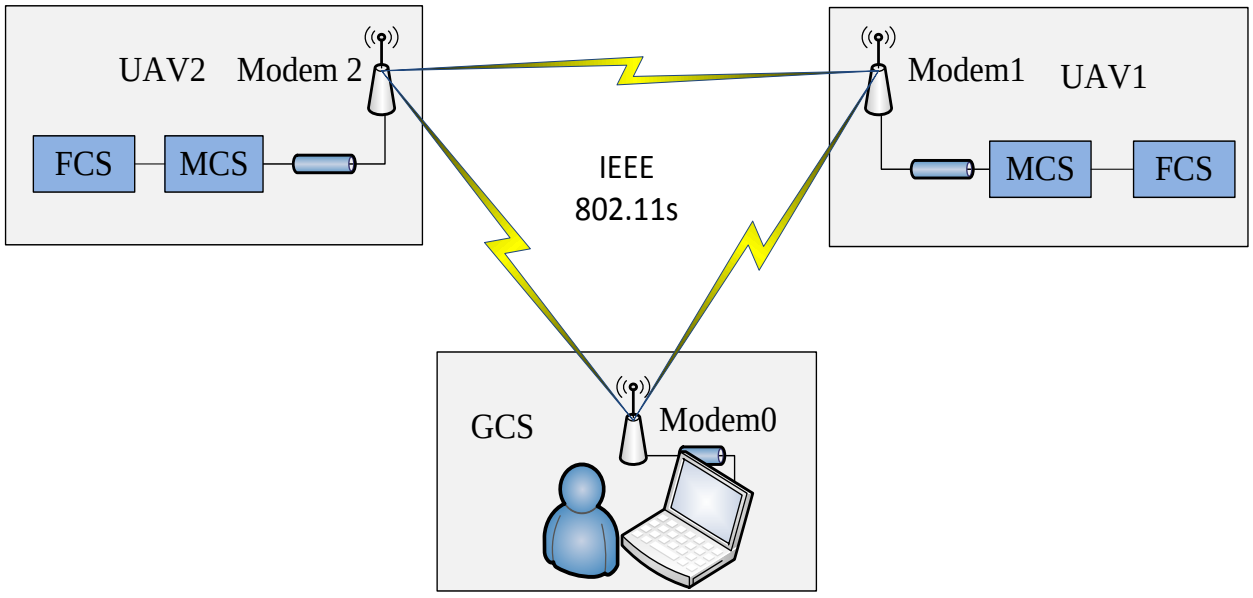

Fig 3.6. Collaborative UAV system network architecture

3.3.1 Neighbor Discovery

A *"hello"* packet containing the UAV's identity is broadcasted at a default frequency of 2 Hz. The default broadcast frequency was determined based on the mobility of the planes. The maximum speed that the gliders can fly with stability is 40 m/s. So, it is fair to assume that the UAV does stay in the neighborhood for at least 0.5 sec assuming the range of wireless modem is at least half kilometer. A list of the neighboring planes in the network is sent in a VACS message to the FCS in regular intervals. The neighbor discovery messages include a list of all the planes in the network and a list of the planes that are one hop away.

Table 3.2. Layout of hello packet

ID	Source Tail	Source MAC	Time

3.3.2 Unicast and Broadcast Message Delivery

Each plane in the CUS is assigned a unique identity number. The FCS send messages with the source field set to its own identity number and destination field set to the identity number of the destination UAV. The user-space application maps the internet protocol (IP) address of the destination based on the identity number of the plane and unicasts a UDP packet to the destination. A packet received from the FCS with a destination identity number of 255 is broadcasted to all the modems by sending the UDP packet to the broadcast address defined in the subnet.

The application was designed such that it could easily be loaded into the RAMS simulator [8] for faster development and validation purposes. The tasks executed by the application can be grouped into four main units. A brief explanation of each unit is given as follows.

3.3.3 Communication Unit

The communication with the FCS and other wireless modems in the network is handled by the communication unit. Two sockets are created for handling communication with neighboring modems and for direct communication with the FCS. The received packets are asynchronously processed in two separate Portable Operating System Interface (POSIX) listener threads. The messages sent and received by the modem are processed by the VACS parser. The parser is implemented as a state machine with each state being a field in the VACS header. The parsed messages are processed based on the message identity number.

3.3.4 Performance and Statistics Unit

The wireless statistics are extracted by parsing the *iw station* and *mpath* dumps. The *iw* commands are executed using the *popen* standard C library function. VACS messages are composed out of the parsed text and sent over to the GCS periodically. The messages sent from each modem to the GCS have to be synchronized in order to facilitate correct post analysis. Therefore, a simple time synchronization mechanism is implemented in which each modem is synchronized with the time of the GCS modem. Every modem sets the current system time in the time field of the "hello" packet. When the "hello" packet is processed the modems update their local time variable to the system time of the GCS modem.

3.3.5 Self-configuration Unit

A table of the IP addresses mapped to the identity numbers of the discovered UAVs in the network is maintained by the self-configuration unit. The map is updated regularly by the main thread and inactive neighbors are removed from the map. A neighboring plane is declared as inactive if the difference between the time-stamp of the last received "hello" packet and the current system time is greater than a defined timeout period. The default period is set to 2 sec.

The IP address to identity numbers map is implemented as a linked list with concurrency support. Access to the linked list is restricted to a single writer thread at any given time. However, multiple reader threads can access the map simultaneously. This mechanism was implemented using read-write locks from pthreads library. The read-write locks were chosen over mutex locks because of the presence of only one writer thread and multiple reader threads in the application.

3.4 Bugs Fixed and Problems Faced

The two main problems encountered implementing the 802.11s-based MANET were kernel panics and system freezes. Kernel panics occurred because of link failures. Further investigation indicated that the HWMP protocol sends a path error frame, when a link failure occurred during the communication. One of the fields in the path error frame was not being appropriately updated which resulted in a null pointer dereference. The kernel panic message was detected when debugging the kernel in real-time over a serial interface in the lab. Most routers typically have 16-32 MB of RAM so it is not feasible to log debugging messages from the kernel or any user-space application. Hence, reproducing and solving problems in the lab environment is very difficult.

The second problem encountered was with the Ethernet interface of the modem. The interface would quickly run out of memory during the start-up and freeze in the 100baseT mode. In order to solve this problem, the Ethernet was switched over to 10baseT mode. Since, the wireless bandwidth is not more than 10 Mbps, the required change should not hinder any wireless performance. After fixing these issues, three 24 hour tests were conducted in the lab to check the up-times of the modems and the results showed that the modems were stable without any freezes or crashes.

The problems were fixed by using Linux kernel debugger tools such as *kgdb* and *ksymoops*. In order to enable kernel debugging over a console, a serial driver called *kgdboc* was installed in Linux.

Chapter 4: Simulation Framework

The RAMS simulator can be extended to simulate advanced networking capabilities by using two different approaches. First, model the networking protocols used in the test-bed from scratch and develop a layered architecture in the network module of the RAMS simulator. Second, use an existing state-of-the-art network simulator and integrate it with the RAMS simulator. The second approach was chosen because of the scope for continuous collaboration and contributions possible from peer researchers around the world to the RAMS as well as the network simulator.

The ns-3 simulator was chosen because of its modular architecture and the support of a wide range of ad-hoc networking protocols. The integration of ns-3 with RAMS simulator involved a couple of design decisions to be taken. An approach for establishing communication and maintaining synchronization between the two simulators has to be chosen. The following are some of the practical ways of integrating the RAMS with a network simulator. One, create a TUN/TAP device in the host operating system and bridge the network interface of ns-3 node with the host interface. This involves running a virtual machine for each node and hence would not be a scalable approach because the virtual machines consume large amounts of host operating system resources. However, for small number of nodes it has an advantage that an ns-3 nodes can act as a real-world networked computer and interact with other systems on the internet seamlessly. Two, merge the event queues of both the simulators together. This requires a lot of work for modification of the simulation core of the two simulators. Moreover, modification of the network simulator would lead to maintaining a separate version of the network simulator for RAMS. This approach would not leverage the benefits and contributions from the large user

community of ns-3. Third, use a socket based interface between the two simulators in order to facilitate the exchange of messages.

The third approach was chosen considering disadvantages of the prior two. Also, in the third approach UNIX Domain Socket (UDS) were preferred over User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) sockets because of the overhead involved in using TCP or UDP messages. The UDS messages unlike TCP and UDP need not be tagged with the transport and network layer headers causing unnecessary overhead in the communication. Furthermore, sending and receiving of UDP and TCP packets would involve more context switches in the operating system causing a decrease in the throughput of the communication.

The following sub-sections describe the simulation framework developed based on RAMS and ns-3 in order to effectively simulate a mobile ad-hoc network of collaborative UAS.

4.1 Architecture

Fig 4.1 shows the architecture of ns-3 integrated with RAMS simulator. The network module of RAMS was modified to incorporate the ns-3 simulator into the RAMS simulator. The network module was split into low-fidelity and high-fidelity sub-components. All the functionality needed to perform the low-fidelity wireless network simulation was encompassed in the low-fidelity sub-component. The high-fidelity sub-component executes the tasks required to establish the communication between ns-3 and RAMS simulator.

The application traffic generated in RAMS is sent back and forth between the two simulators through UDS sockets. The VACS packets generated by the agent or MCS module are captured by the high-fidelity network module, processed and then sent over to the ns-3 node. The application layer of the ns-3 node parses the VACS packet, extracts the destination node

information, and sends the packet to the destination by using the socket programming interface provided by the ns-3 simulator.

The network module updates the position of the nodes in ns-3 by sending mobility information on every iteration of the module. The position information in RAMS is translated by the network module from GPS latitude, longitude and altitude to Cartesian coordinates before sending over to the ns-3. The transformation from GPS to Cartesian coordinate system is performed by considering the GCS location to be at origin. The ns-3 controller receives the mobility information from the network module of RAMS simulator through a unidirectional UNIX pipe.

The virtual network of nodes is created by the ns-3 controller. Based on the type of the network being configured, the necessary protocol stacks are installed on each node. Two types of network architectures are provided by the simulator namely, infrastructure, and mesh. In the infrastructure mode, the GCS is configured as an access point and rest of the nodes as Wi-Fi stations. In the mesh mode, all the nodes are configured as IEEE 802.11s mesh nodes and every node can thus send, receive and forward packets. Classes for instantiating a Wi-Fi- or mesh-network are derived from the *network* base class. The base class implements general functionality for adding, and deleting nodes, printing statistics, and updating position and attitude of the node. The derived classes, *wifi* and *mesh*, create the specific network architecture and install all the necessary protocol stacks.

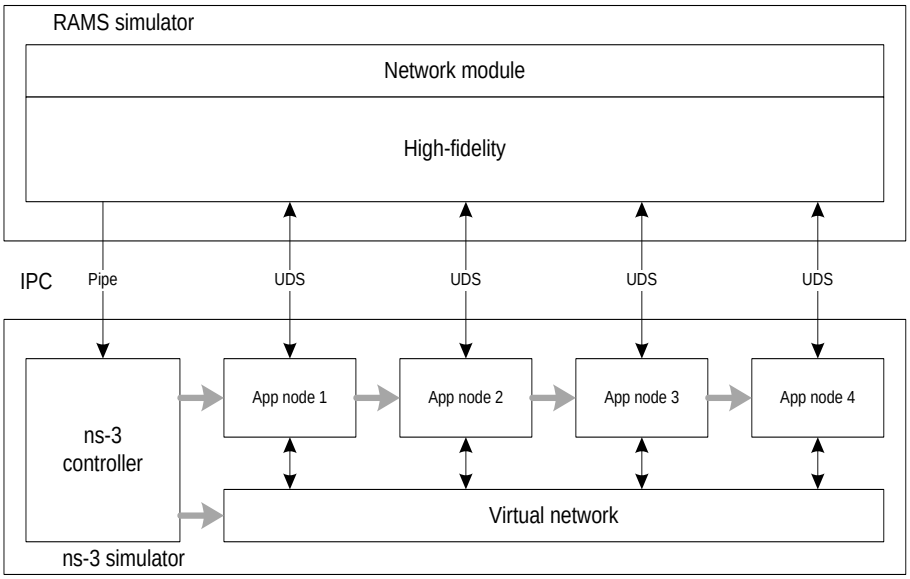

Fig 4.1. Architecture of simulation framework based on RAMS and ns-3

The two processes register signal handlers to handle the events raised by each other. Some of the events of interest include the ns-3 signaling start and stop of the simulation. The ns-3 raises SIGUSR1 signal during the start of simulation and issues SIGUSR2 when the simulation is complete. The stop time of the network simulation is set by the RAMS during the configuration step. The signal handler of network module processes the SIGUSR2 signal and executes all the stop routines of the individual modules and generates mobility and wireless statistics. The graceful termination of processes is required in order to extract statistics from both the simulators at the end of simulation.

The data-link and physical layers employed in the simulation use *yans-wifi* modules for the simulation of a Wi-Fi modem. The Wi-Fi model is used to closely resemble the real systems. The routing of packets in the simulation is carried out in the data-link layer by utilizing an implementation of Hybrid Wireless Mesh Protocol (HWMP) defined in the IEEE 802.11s mesh networking standard.

4.2 Application Layer

The application layer is extended to provide a wrapper framework for running user-space code that normally runs on a physical wireless modem. The applications of interest function as a communication interface between autonomous navigation systems. The wrapper framework facilitates in loading the user-space code dynamically during the run time of simulation. The wrapper replaces the hardware system calls made by the application to send and receive packets with the functions designed to establish communication with the application layer of ns-3.

The UDS file descriptors associated with each node in ns-3 are directly passed to the user-space code, so that any VACS packets sent from the RAMS simulator will directly be processed in the user-space code. The wrapper only adds the IP address of the source node to the packet, in order for the user-space code to create a map between tail number and IP address. The packets on both the sides are parsed in order to extract the IP addresses of the source and destination nodes. The UDP packet is sent to the IP address of the destination node in ns-3 by calling the corresponding send function in the ns-3 socket library.

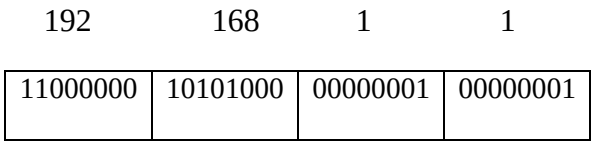

Fig 4.2. UDS Header

The dot-decimal notation of the IP address is encoded as a four byte value and stored in the header as four binary numbers. That is, each decimal number is converted to its binary form and stored in a byte wide memory as shown in Fig 4.2.

As an alternative to using the dynamically loaded user-space code a simple pass-through application is developed providing unicasting and broadcasting services. The IP address to tail number translation is taken care by the application. The pass-through application is implemented in a class called *passTruApp* derived from the Application class of ns-3.

4.3 Propagation Loss Model

The log-distance propagation loss model in ns-3 was used to model the propagation losses that occur in a UAV to ground or UAV to UAV channel. The path loss equation for the propagation loss model is given by the following the equation:

$$\text{L (dB)} = L_0 + 10n \log_{10}\left(\frac{d}{d_0}\right)$$

Where, $L_0$is the path loss in dB occurred at a reference distance of d_0 meters, d is the distance of separation between transmitter and receiver, and n is the path loss distance exponent. The parameters in the path loss model that can be tuned are the path loss exponent, the reference distance, and the reference path loss.

Several simulations were performed to tune the model so that the path losses calculated in the simulator are close to the losses that occur in a real world test-bed. The values of parameters obtained for the close resemblance and least error are shown in the Table 5.3 of RSSI Validation section.

4.4 Mobility Model

The user-driven mobility model currently available in RAMS was extended with random waypoint sphere, random waypoint cube, and trace driven mobility models to accelerate the experimentation process. The random waypoint models assign waypoints to each plane using

Monte Carlo methods. After a plane arrives close to the target waypoint, a new waypoint will be generated and assigned to the plane. Once a waypoint is assigned to an agent module of RAMS, it takes care of the target heading and bearing in order to navigate to the target waypoint. The following sub-sections describe the three mobility models that were designed and implemented in the RAMS simulator.

Random waypoint sphere mobility model generates waypoints in a sphere of given radius. The GPS coordinates of the center and radius of the sphere are given as inputs to this model. A waypoint is defined by a set of latitude, longitude and altitude coordinates and is used to represent a point in physical space. The GCS is assumed to be at the center of sphere. In order to calculate a target waypoint which is at a distance d from the reference location (GCS), first the latitude and longitude of the target waypoint are calculated. The target latitude and longitude can be calculated by the following equations [30]:

$$Target\ Latitude = \sin^{-1}(\sin(lat)\cos(d) + \cos(lat)\sin(dist)\cos(bearing))$$

$$Target\ Longitude = lon + \tan^{-1}\left(\frac{\sin(bearing)\sin(d)\cos(lat)}{\cos(d) - \sin(lat)\sin(target\_lat)}\right)$$

Where, $bearing$ is the angle made by the target waypoint with the reference north pole, lat is the latitude of the GCS location, lon is the longitude of the GCS location. The value of $bearing$ is generated by a random number generator between the range of 0 and 360 degrees. The distance d is also a random number which varies between 0 and the radius of the sphere. Given the radius of the sphere and distance of the target waypoint from center, the maximum altitude can be calculated by Pythagoras formula. Another random number is generated for the target

altitude in the range of 0 and maximum altitude. The waypoints generated are always in the upper hemisphere by considering the GCS to be on the ground at zero altitude.

The random waypoint cube model can be used when the boundaries of the flying area are known. The maximum and minimum latitude, longitude and altitude values are given as inputs and random waypoints are generated by the model.

The trace driven mobility model parses a file containing latitude, longitude and altitude information of the waypoints and feeds it to the agent module. The trace can either be collected from the real-world flights or from the simulation using other mobility models.

4.5 Configuration and Logging

Some of the variables in simulation are made configurable at the launch of the simulator so that the unnecessary compilations of code are avoided. This feature enables running several experiments in a simple bash script and accelerates the experimentation process. The following variables in the simulator are made configurable.

1. Radius: Radius of the sphere given as an input to the random waypoint sphere mobility model
2. ns3-timeout: The time in seconds given to set the HWMP active path timeout value
3. ns3-retries: The number of PREQ retries made in the HWMP protocol
4. ns3-queueSize: The queue size of the HWMP protocol
5. ns3-connPerNode: The number of UDP connections per node that need to generated during the simulation time
6. ns3-stopTime: The simulation stop time of the ns-3 simulator
7. plane-count: The number of planes to be created for the simulation

8. node-speed: The speed of the planes measured in knots/sec

Conventional approaches to collect the necessary information from simulators, print the necessary information with key delimiters so that they can be processed after the simulation by scripts written in PERL or python scripting languages. The parsers written with simple delimiters and keywords are not robust because of colliding keywords or delimiters. This problem can be alleviated by creating logs based on XML. All the logging is done inside the “ns3” tag. The attributes of “ns3” are the parameters of the simulation mentioned above so that the files can be easily distinguished during the post-processing step. The parser written in PERL reads all the XML files from the simulator log directory, parses them, and generates statistics like throughput, packet loss percentage, delays, and other metrics of interest.

Chapter 5: Results and Discussions

This chapter presents the results obtained from real-world experiments and simulations. The experiments were conducted to validate the open80211s implementation, verify and validate the simulation framework, and evaluate the performance and reliability of the IEEE 802.11s standard.

The metrics of interest in the following experiments include throughput and Packet Loss Percentage (PLP). Throughput is defined as the ratio of total number of bits transmitted to the time difference between first bit and last bit arrived at the receiver during a communication session as shown in the following equation.

$$\text{Throughput (bps)} = \frac{n}{t_{last} - t_{first}}$$

Where, n is the number of bits arrived at the receiver during a communication session, and t_{first} and t_{last} are the times at which the first and last bits arrived at the receiver, respectively. Throughput indicates the performance of a wireless link, that is, a high performance link allows to send data at higher rates. PLP is defined as the ratio of total number of lost packets to the total number of the packets transmitted as shown in the following equation.

$$PLP = \frac{n_{transmitted} - n_{received}}{n_{transmitted}}$$

Where, $n_{transmitted}$ is the total number of packets transmitted, and $n_{received}$ is the total number of packets received during a communication session. PLP indicates the reliability of a wireless link, that is, a link with low PLP value is more reliable.

5.1 Open80211s Validation

The following experiment was designed to validate the open80211s implementation and to study the reconfigurability of the IEEE 802.11s mesh network. The experimental scenario consisted of two UAVs and a GCS. The flight paths of the two UAVs, and GCS location is shown in Fig 5.1 & Fig 5.2. The parameters of the experiment are listed in Table 5.1.

Table 5.1. Values of parameters involved in the experiment

Parameters	Values
UAV air-speed	15 m/sec
Altitude	150 meters
TX power	17 dBm
Network architecture	Mesh
HWMP active path timeout	5.12 sec
Iperf load	500 Kbps
Iperf packet size	1470 bytes

Fig 5.1. UAV1 flight path, GCS location is marked by a black star

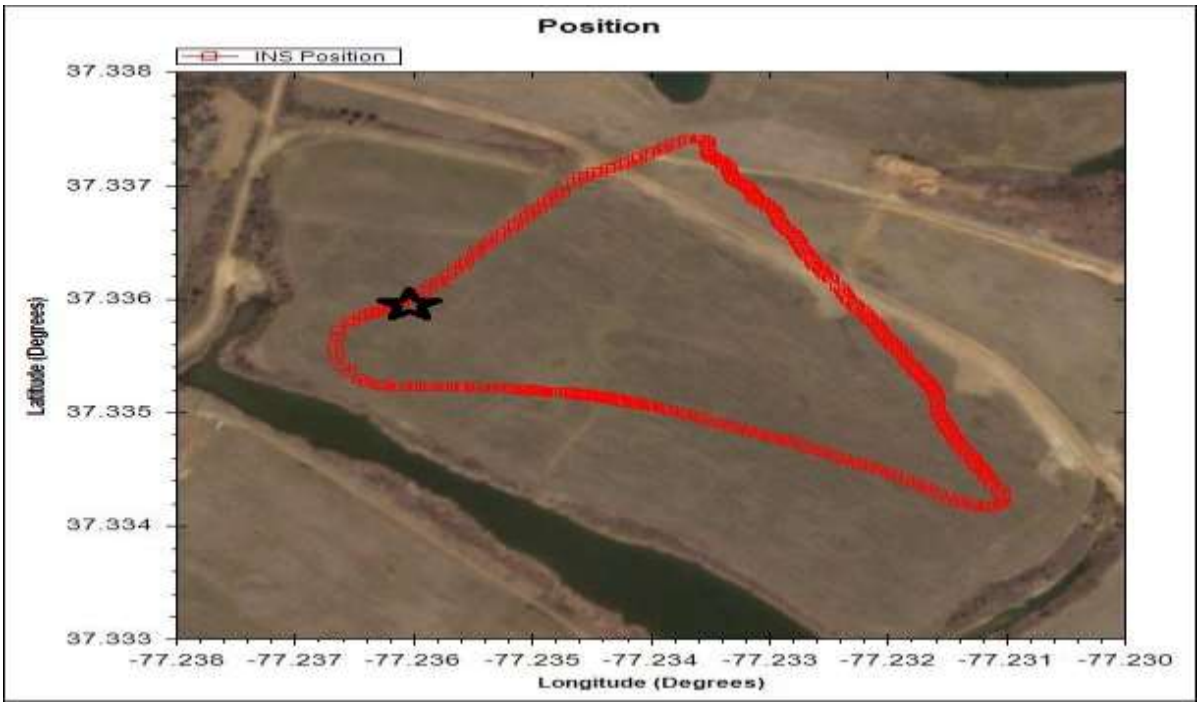

Fig 5.2. UAV2 flight path, GCS location is marked by a black star

The percentage of lost packets was measured by using the *iperf* network performance monitoring tool once for every two seconds. The number of frame errors, frame retries, and the

Received Signal Strength Indicator (RSSI) values were obtained from *iw station* and *mpath* dumps at a frequency of 1 Hz. During the experiment two *iperf* servers were installed on UAV1 and GCS nodes. At the same time, two *iperf* clients were installed on UAV1 and UAV2, creating network flows from UAV1 to GCS and from UAV2 to UAV1.

In Fig 5.3 the percentage of lost UDP packets and number of hops is plotted over time to show the impact of packet losses on reconfigurability of the network. This data is corresponding to the network flow from UAV2 to UAV1. The existence of a direct link between an *iperf* server and client is indicated by a value of 1 for the number of hops in all the figures. A hop value of 2 means a node in between the client and server is relaying the messages to establish the network flow. It is clear that the network is very sensitive to any packets losses in the network. The network reconfigures at time instants 19, 23, 30, 34, 40, 55, 59, and 64 sec. At each of these time instants packet loss occurs and the network tries to find a better route. For instance at time 18 seconds, the observed packet loss was 21%, due to which the UAV2 hops over GCS to send data to UAV1. Again at time 24 sec, the observed packet loss was 12% due to which the UAV2 now sends data directly to UAV1 instead of hopping over GCS. So, the HWMP protocol created new paths dynamically every time packet losses occurred on the existing path.

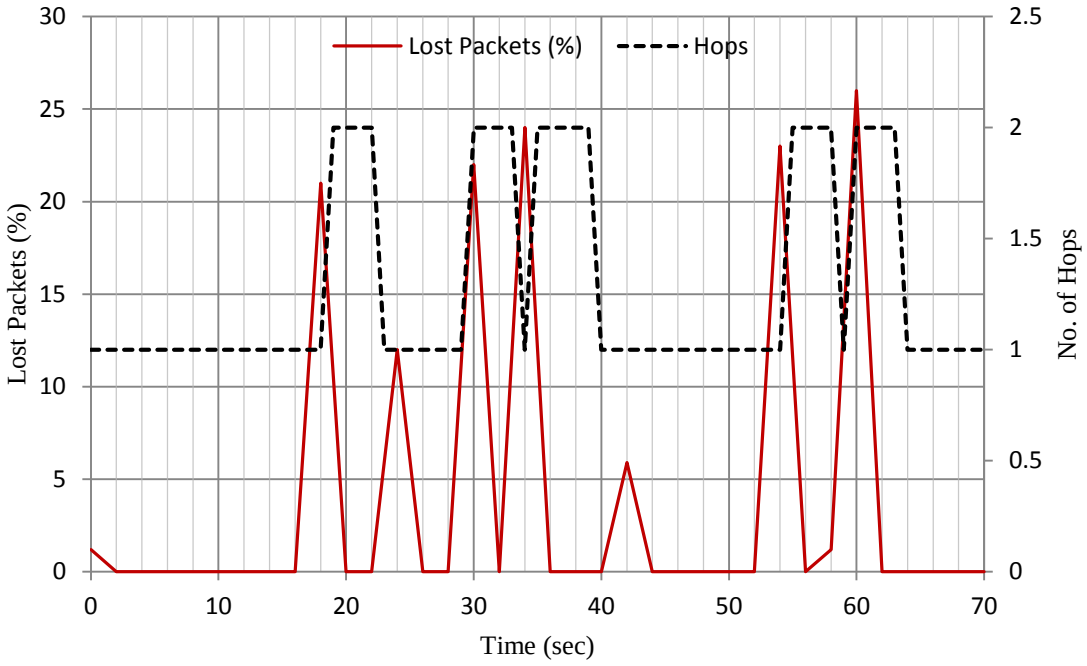

Fig 5.3. Impact of UDP packet loss on reconfigurability

Each UDP packet can either be transmitted in a single frame or multiple frames in the data-link layer based on the size of the packet. Thus, the frames lost in the data-link layer will always be greater than or equal to the number of UDP packets lost. The frame errors as shown in Fig 5.4 occurred at the same time instants as the UDP packet losses occurred in Fig 5.3.

A frame could be retransmitted a maximum number of times as set in MAC layer's parameters before it is discarded. Thus, the number of frame retransmissions in Fig 5.5 hits peaks corresponding to the time instants where frame errors occurred. For the same reason the effect of frame retransmissions was not as profound as frame errors on the reconfigurability of the network.

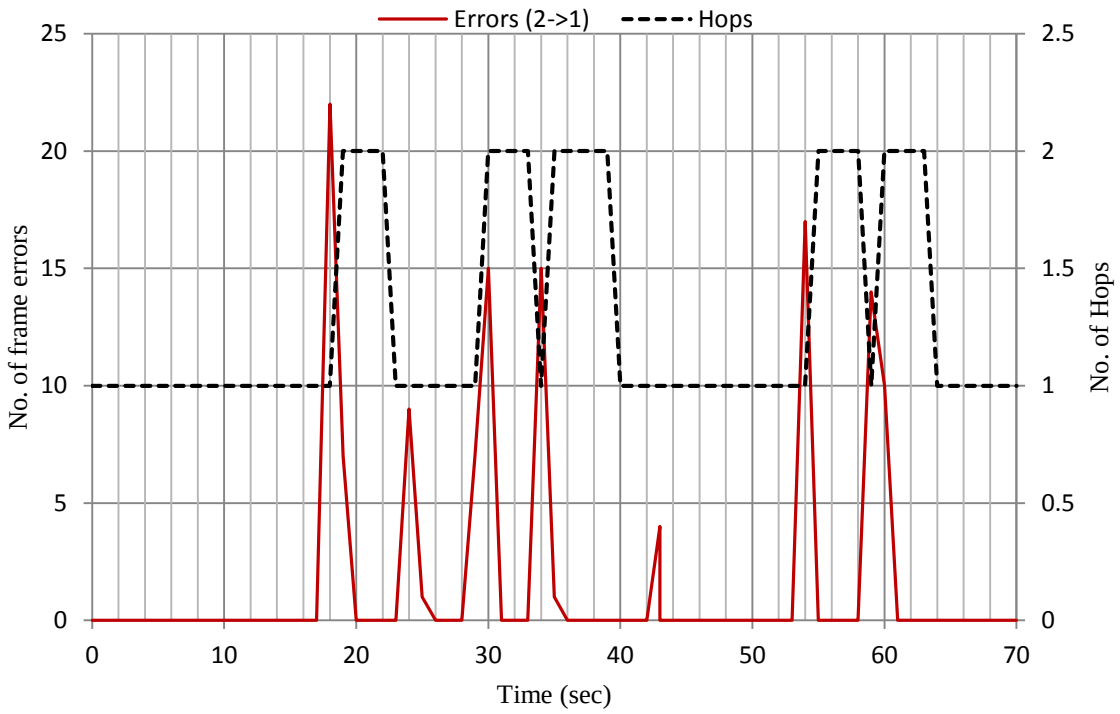

Fig 5.4. Impact of frame errors on reconfigurability

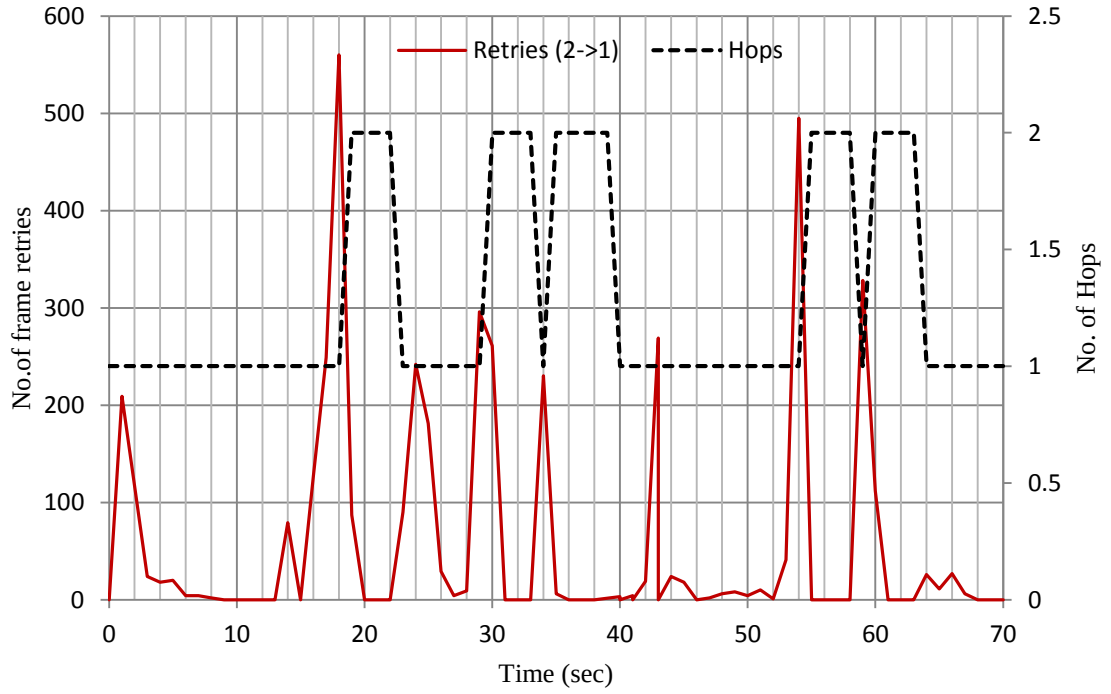

Fig 5.5. Impact of frame retries on reconfigurability

Fig 5.6 shows the impact of RSSI on reconfigurability of the wireless network. RSSI (2->0) indicates the strength of the received signal at the GCS from UAV2 and RSSI (2->1) indicates the strength of the received signal at UAV1 from UAV2. This data is corresponding to the previous mentioned network flows created by the *iperf* servers and clients. At the time instants that the network reconfigures itself, the RSSI on the existing links drops below the RSSI of an alternate link. As the RSSI goes down the packet losses can be observed at the data-link and transport layer. The RSSI can therefore be used as an estimate for link quality.

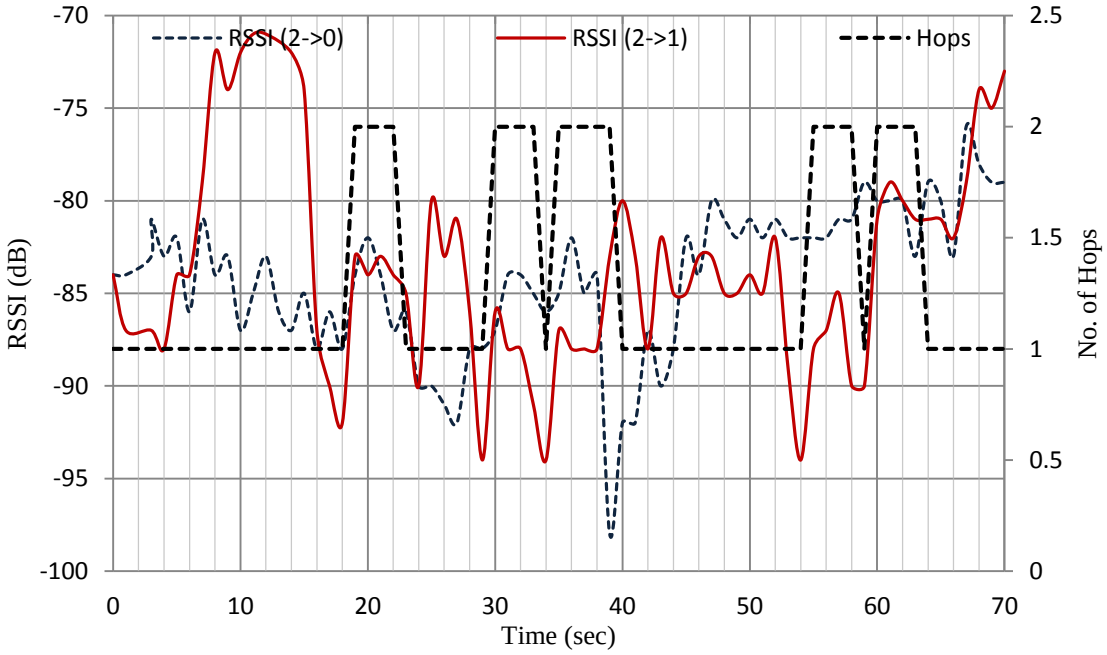

Fig 5.6. Impact of RSSI on reconfigurability

5.2 Simulator Validation

This section describes the experiments conducted to validate the integration of RAMS with ns-3 simulator, and to validate the propagation loss model used in the simulation.

5.2.1 Mobility Validation

This section describes the validation of integrating the RAMS with the ns-3 simulator. In the

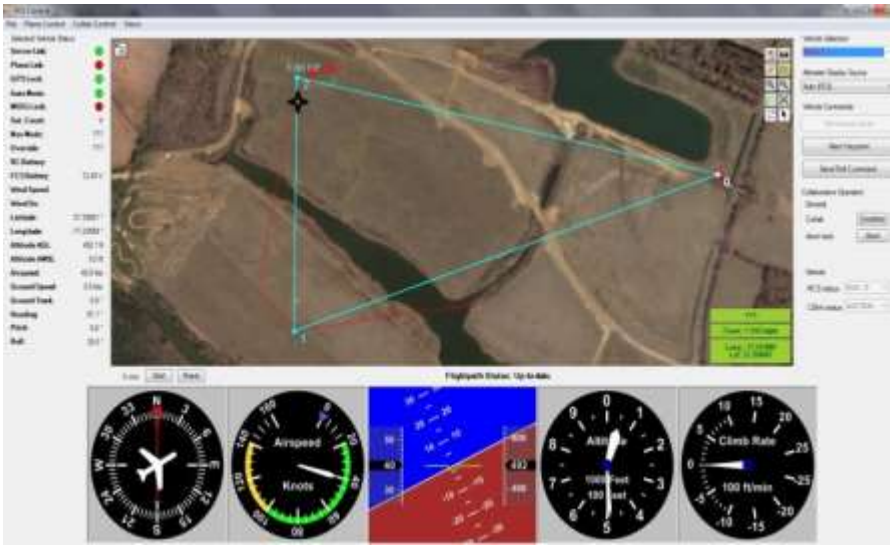

Fig 5.7. GCS user-interface and flight path of UAV

test scenario, the GCS is marked by a black star in Fig 5.7, the UAV is marked in red color, and the waypoints assigned to the plane are denoted by 0, 1, and 2 in blue. The plane starts flying from the GCS location and traverses waypoints 0, 1 and 2 subsequently. The packet drops based on packet error rate calculation is disabled in this experiment, in order to focus on the packet drops caused solely by distance of separation between nodes. The parameters of simulation are listed in Table 5.2.

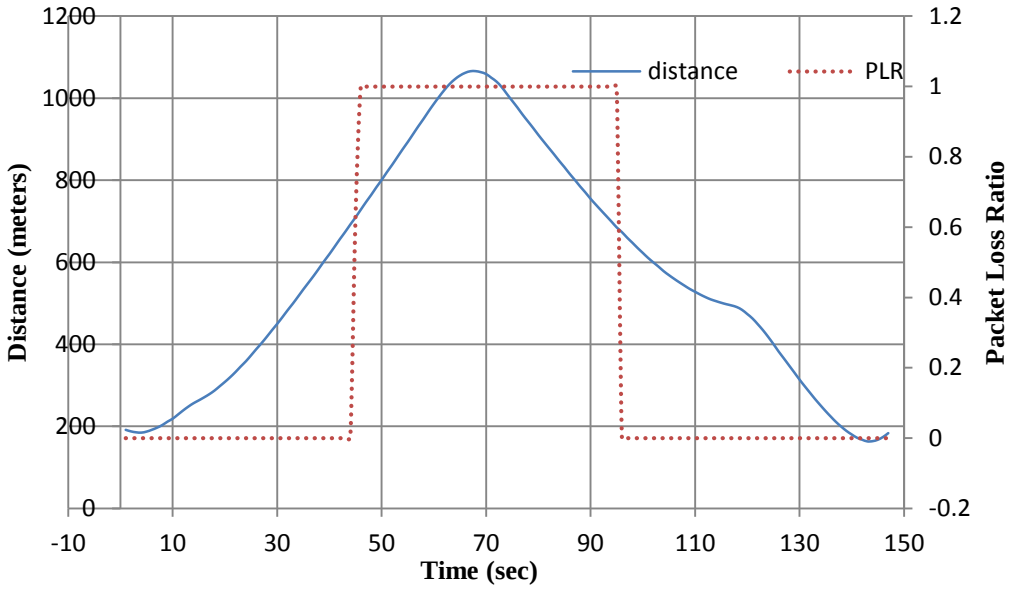

Fig 5.8. Impact of distance of separation between nodes on packet loss ratio

In Fig 5.8 the separation distance and Packet Loss Ratio (PLR) is shown, verifying correct integration of RAMS with ns-3. When the separation distance increases the Received Signal Strength (RSS) reduces. Subsequently packets are being dropped when the RSS level falls below a certain threshold resulting in a link failure. The Packet Loss Ratio (PLR) is the ratio of number of lost packets to the total number of transmitted packets. The purpose of this experiment is to validate that the mobility model is correctly integrated with the ns-3 simulator.

Table 5.2. Simulation parameters

Parameters	Values
Transmit power	17 dBm
Buffer size	255
TX bitrate	6 Mbps
HWMP PREQ retries	3
HWMP active path timeout	5.12 sec

5.2.2 RSSI Validation

The experiments conducted to tune the propagation loss model involved a single UAV and one GCS node. Same waypoints were assigned to the UAV both in the simulation as well as real-world. The RSSI values were measured as the UAV flies from waypoint 0 to 1, 2, 3 and back to 0. The GCS node is marked with black color in Fig 5.9. Average speed of the UAV was approximately 15 m/sec and the transmit power of the modems was set to 17 dBm both in simulation and real-world.

Each Received Signal Strength Indicator (RSSI) sample obtained in the simulation corresponds to an RSSI value calculated for a packet received at the GCS node. Whereas, in the real-world experiments, the RSSI at the GCS node was recorded for every two seconds. An RSSI value that is closer to zero means better signal. Fig 5.10 and Fig 5.11 show the plots of RSSI obtained from the real-world and simulation platforms. The RSSI in Fig 5.11 varied from -75 to -93 dB whereas in Fig 5.10 it varied from -80 to -92 dB. The average error occurred in calculating the RSSI by the propagation loss model was 7.6 dBm. Standard deviation of the RSSI data obtained from real-world test-bed experiments was 11.7 and simulation was 3.67. The

transmission range was found to be 769 meters by using the propagation loss parameters of Table 5.3, physical layer parameters of Table 5.5, and TX bit rate of 6 Mbps.

Fig 5.9. The set of waypoints 0, 1, 2 and 3 (red) assigned to the UAV in the real-world and simulation to validate the propagation loss model

Table 5.3: The values of parameters in Log distance propagation loss model

Parameters	Value
Reference loss	-53 dBm
Reference distance	1 meter
Exponent	2.0

Table 5.4. Comparing RSSI of simulator and test-bed

Platform	Mean RSSI (dBm)	Standard Deviation
Test-bed	-80.3	11.7
Simulation	-87.9	3.67

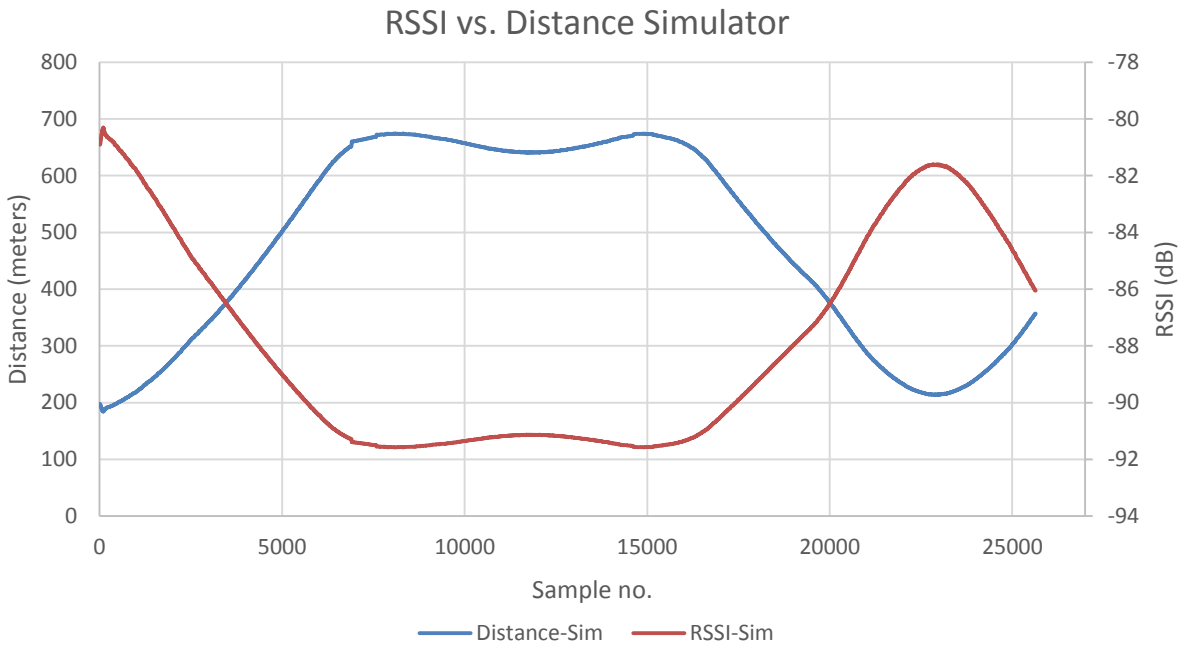

Fig 5.10 A plot of RSSI vs. distance using Log distance propagation loss model

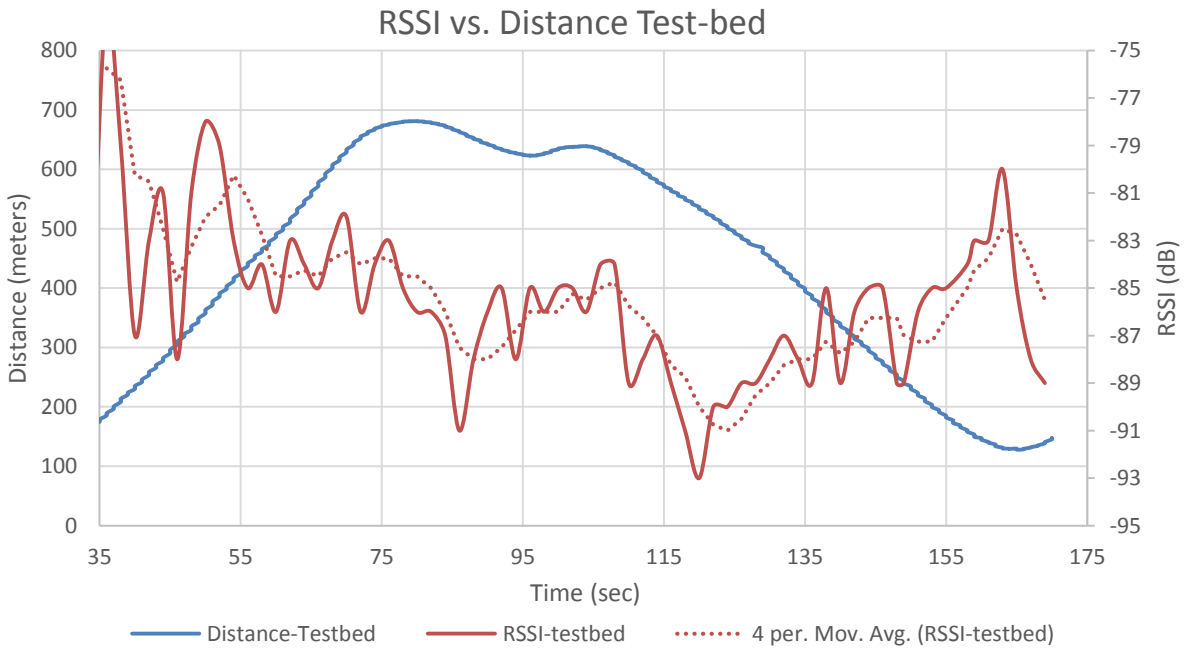

Fig 5.11. A plot of RSSI vs. distance in real-world test-bed

TX power levels are the number of power levels available between TX power start and end. TX power end is the maximum available transmission power level and TX power start is the minimum. By setting TX power levels to 1 and TX Power start and end to 17 dBm, the device is set to always transmit at a power of 17 dBm. The energy of the received signal needs to be higher than the energy-detection-threshold (dBm) to allow physical layer to detect the signal.

Table 5.5. PHY layer parameters

Parameter	Value
Energy detection threshold	-96 dBm
TX gain	1 dB
RX gain	1 dB
Noise	7 dB
TX power levels	1
TX power start	17 dBm
TX power end	17 dBm

5.3 Performance Evaluation

This section evaluates the mobile ad-hoc network of UAVs in the real-world test-bed developed as part of this research.

5.3.1 Impact of hop count

The goal of the following experiments was to study the impact of number of hops on metrics of the network namely, throughput, and PLP. Three real-world scenarios namely, one hop, two hop, and three hop, were designed to evaluate the multi-hop capabilities of the IEEE 802.11s. In each scenario, the network was evaluated by varying the load offered from 500 Kbps to 9 Mbps. The scenarios consist of wireless modems and two end-systems (laptops). Henceforth, the wireless modems will be referred to as nodes. The one hop scenario consist of two end systems, and two nodes sharing a wireless link, as shown in Fig 5.12. In the 2 hop scenario, three nodes were placed such that, nodes A and C do not share a direct wireless link and node B is in the range of both the nodes as shown in Fig 5.13. In the 3 hop scenario, four

nodes were placed such that only the nodes A and B, B and C, and C and D share a direct link as shown in

Fig 5.14. The nodes in the experiments were placed such that the average RSSI between any two adjacent nodes is around -80 dBm at around 15 meters indoors. The throughput for a given load was measured by the *iperf* network performance tool. The clients and servers of the *iperf* were run on the end systems which are connected to the wireless modems. In all these experiments the laptop connected to A was the server. Each experiment was one min long and the values of throughput and PLP were the average values obtained during one minute.

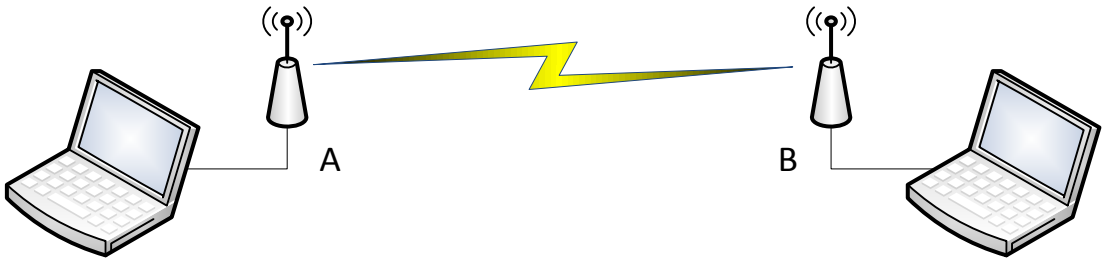

Fig 5.12. One hop scenario

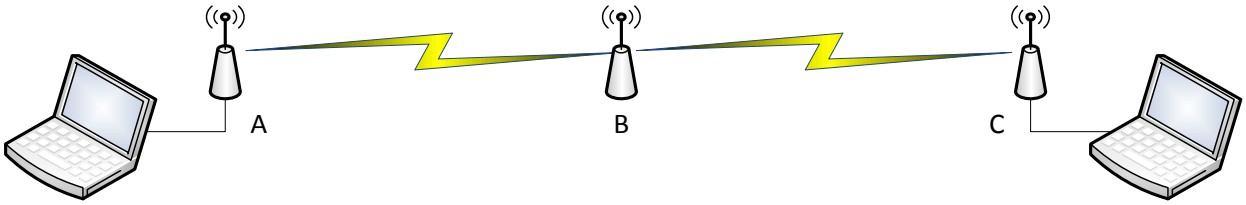

Fig 5.13. Two hop scenario

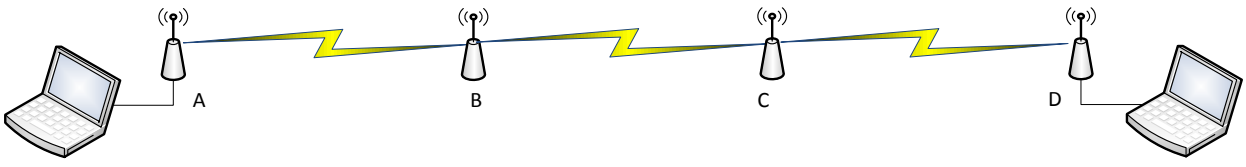

Fig 5.14. Three hop scenario

Table 5.6. Test-bed parameters in the experiment

Parameters	Values
TX power	17 dBm
Network architecture	Mesh
HWMP active path timeout	5.12 sec
Iperf packet size	1470 bytes
Average RSSI	-80 dBm
PHY TX bit-rate	6 Mbps
Experiment duration	1 min
Distance of separation	15 meters
Environment	Indoors

From Fig 5.15 we can observe that the PLP for 1 hop scenario stays less than 2% until the load increases to 5 Mbps and from then on the PLP increases 10% for every 1 Mbps increase in load. In a wireless modem, the packets that are ready for transmission are queued in packet buffer. The packet count in the buffer grows when the rate of incoming packets exceeds the rate of outgoing packets. That is, the number of packets that get queued increase as load increases beyond the wireless bandwidth, 5 Mbps, in this case. The increase in PLP is, therefore, due to packets getting dropped at the queue because of lack of space in the buffer. As the number of hops increase the PLP curve gets steeper due to increase in probability of packet errors. The reason for increase in probability of packet errors with increase in number of hops/links in a path is demonstrated in the following example. If the probability that a packet gets successfully

received is 0.9 on link1 and 0.8 on link2, then the probability that the packet arrives at the destination successfully on the path containing link1 and link2 is 0.9*0.9, that is, 0.81. Therefore the loss probability goes up from 0.1 to 0.19 due to increase in number of hops/links in the path.

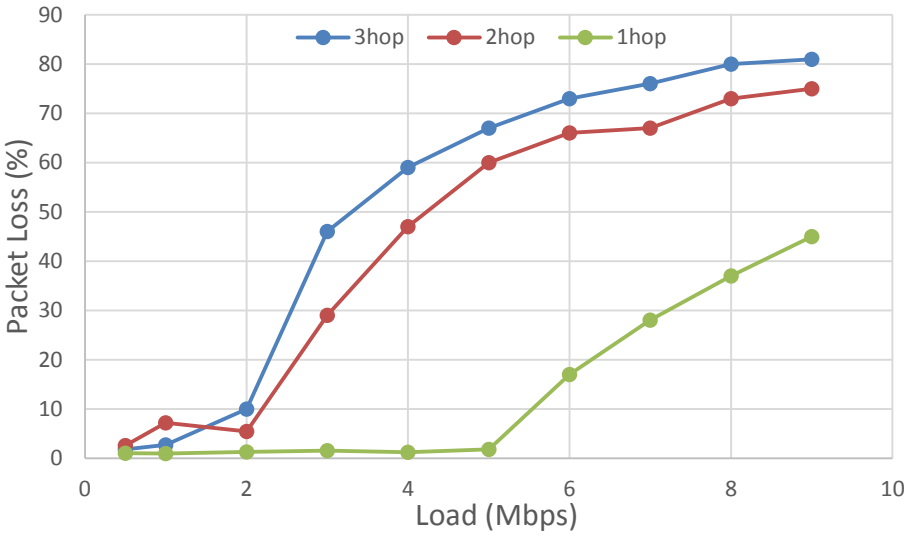

Fig 5.15. Impact of number of hops on packet loss percentage

From Fig 5.16 we can observe that the throughput increases with the offered load initially until the maximum bandwidth available is consumed and from then it almost remains constant. The maximum possible throughput decreases with the increase in number of hops. Throughput is directly proportional to the number of data bits received and inversely proportional to the duration of transmission from the equation described before. In this case, the decrease in throughput can be attributed to the decrease in number of data bits received at the destination, that is, due to the increase in packet losses as shown in Fig 5.15. A minor contribution to decrease in throughput comes from the increase of total transmission time, sum of transmission times at each node in the path, as the number of hops increase, causing an increase in end-to-end delay.

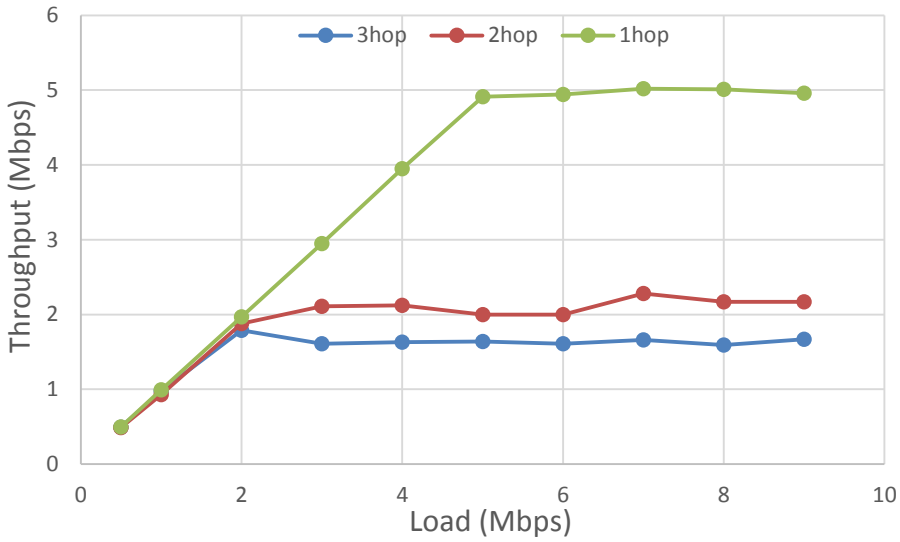

Fig 5.16. Impact of number of hops on throughput

Chapter 6: Conclusions and Future Work

6.1 Conclusion

A reconfigurable network based on the IEEE 802.11s mesh networking standard was developed for collaborative UAV applications. The development efforts include implementation of a user-space application on the wireless modem, building a custom Linux kernel for the wireless modem, identifying and fixing the issues in the open80211s package, and selecting a wireless modem and antenna. The functionality of the developed network was successfully demonstrated in a scenario with two UAVs and a GCS.

The reference implementation of the IEEE 802.11s mesh networking standard called open80211s was validated and an initial evaluation was carried out in a mobile scenario. The impact of UDP packet losses, frame errors, frame retries, and RSSI on the reconfigurability of the network was studied. The results showed that the network is sensitive to UDP packet losses, frames errors, frame retries and variation of RSSI on a link. The UDP packet losses and frame errors caused an immediate reconfiguration of the wireless network while the RSSI changes and frame retries on a link did not always cause the network to reconfigure and contribute to a more reliable and stable link. Overall, the results showed that the IEEE 802.11s is a promising solution for collaborative UAV applications.

The performance of the IEEE 802.11s mesh network was evaluated by studying the impact of offered load on throughput and packet loss percentage in 1 hop, 2 hop and 3 hop static, indoor scenarios. The results indicated that the maximum wireless bandwidth was 5.02 Mbps in a 1 hop scenario, 2.28 Mbps in 2 hop scenario, and 1.66 Mbps in a 3 hop scenario. The throughput increased with the offered load until the maximum wireless bandwidth was consumed, and then

remained constant. Packet loss percentage stayed under 2% for offered loads of under 5 Mbps and from then on the packet loss percentage increased approximately by 10% for every 1 Mbps increase in load.

A simulation framework for the simulation of mobile ad-hoc network of UAVs was built based on the RAMS and ns-3 simulators. The application layer of the ns-3 simulator was extended to execute the userspace applications that run on physical wireless modems. The Log distance propagation loss model was tuned to produce RSSI values with an average error of 7.6 dBm compared to real-world. The user-driven mobility model of RAMS was extended with random waypoint and trace-driven models to automate the network performance evaluation process.

6.2 Future Work

New high-fidelity mobility and antenna models need to be added to the simulation framework in order to simulate the effect of aircraft's attitude and antenna radiation patterns on the wireless network. Although, the propagation loss model was tuned to produce close to real-world losses, in the future, more accurate propagation loss models should be derived.

The HWMP protocol can be extensively evaluated by using the simulation framework built in this research. The use of RSSI as an estimate for link quality to route around the link failures should be investigated. The effectiveness of preemptive routing techniques applied to collaborative UAV networks is an interesting next step for further research [31].

The active path timeout parameter of the HWMP protocol impacts throughput by influencing the routing protocol overhead. A technique to set the timeout parameter dynamically by the protocol needs to be developed so that the protocol overhead is reduced.

Bibliography

[1] Z. Sarris and S. ATLAS, "'Survey of UAV applications in civil markets (June 2001)," *The 9 th IEEE Mediterranean Conference on Control and Automation (MED'01)*.

[2] T. Samad, J.S. Bay and D. Godbole, "'Network-Centric Systems for Military Operations in Urban Terrain: The Role of UAVs," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 92-107.

[3] A. Ryan, M. Zennaro, A. Howell, R. Sengupta and J.K. Hedrick, "'An overview of emerging results in cooperative UAV control," *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, pp. 602-607 Vol.1.

[4] Anonymous "'IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1-2793.

[5] M. Abolhasan, T. Wysocki and E. Dutkiewicz, "'A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 1, 1, pp. 1-22.

[6] M.E.M. Campista, P.M. Esposito, I.M. Moraes, L. Costa, O. Duarte, D.G. Passos, de Albuquerque, Célio Vinicius N, D.C.M. Saade and M.G. Rubinstein, "'Routing metrics and protocols for wireless mesh networks," *Network, IEEE*, vol. 22, no. 1, pp. 6-12.

[7] open80211s, "'http://open80211s.org/open80211s/,".

[8] T. Bakker, G. Ward, S. Patibandla and R.H. Klenke, "'RAMS: A Fast, Low-Fidelity, Multiple Agent Discrete-Event Simulator," *2013 Summer Simulation Multi-Conference (SummerSim'13)*.

[9] ns-3, "'http://www.nsnam.org/,".

[10] G.R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann and B. Walke, "'IEEE 802.11 s: the WLAN mesh standard," *Wireless Communications, IEEE*, vol. 17, no. 1, pp. 104-111.

[11] J. Pan and R. Jain, "'A survey of network simulation tools: Current status and future developments," *Email: jp10@ cse.wustl.edu*.

[12] F.J. Martinez, C.K. Toh, J. Cano, C.T. Calafate and P. Manzoni, "'A survey and comparative study of simulators for vehicular ad hoc networks (VANETs)," *Wireless Communications and Mobile Computing*, vol. 11, no. 7, pp. 813-828.

[13] K. Andreev and P. Boyko, "'IEEE 802.11 s Mesh Networking NS-3 Model," *IITP, WNS3, March*.

[14] M. Esquius Morote, "'IEEE 802.11 s Mesh Networking Evaluation under NS-3,".

[15] L.B. Mize IV, *Development of a Multiple Vehicle Collaborative Unmanned Aerial System*.

[16] J. Ortiz, "'Development of a Low Cost Autopilot System for Unmanned Aerial Vehicles,".

[17] T.X. Brown, B. Argrow, C. Dixon, S. Doshi, R. Thekkekunnel and D. Henkel, "'Ad hoc uav ground network (augnet,".

[18] E.W. Frew and T.X. Brown, "'Airborne communication networks for small unmanned aircraft systems," *Proc IEEE*, vol. 96, no. 12.

[19] S. Morgenthaler, T. Braun, Zhongliang Zhao, T. Staub and M. Anwander, "'UAVNet: A mobile wireless mesh network using Unmanned Aerial Vehicles," *Globecom Workshops (GC Wkshps), 2012 IEEE*, pp. 1603-1608.

[20] P. Pandey, S. Satish, J. Kuri and H. Dagale, "'Design & Implementation of IEEE 802.11 s Mesh Nodes with enhanced features,", pp. 639-644.

[21] OpenWrt, "'https://openwrt.org/,".

[22] Linux Wireless, "'http://wireless.kernel.org/,".

[23] L.J. SánchezCuenca, "'802.11 s based Wireless Mesh Network (WMN) test-bed,".

[24] A. Hasan, B. Pisano, S. Panichsakul, P. Gray, J. Huang, R. Han, D. Lawrence and K. Mohseni, "'Sensorflock: A mobile system of networked micro-air vehicles," *Department of Computer Science University of Colorado at Boulder, Tech.Rep*.

[25] R.G. Garroppo, S. Giordano and L. Tavanti, "'A joint experimental and simulation study of the IEEE 802.11s HWMP protocol and airtime link metric," *Int.J.Commun.Syst.*, vol. 25, no. 2, feb, pp. 92-110.

[26] D. Hague, H.T. Kung and B. Suter, "'Field Experimentation of Cots-Based UAV Networking," *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1-7.

[27] C. Danilov, T.R. Henderson, T. Goff, J.H. Kim, J. Macker, J. Weston, N. Neogi, A. Ortiz and D. Uhlig, "'Experiment and field demonstration of a 802.11-based ground-UAV mobile ad-hoc network," *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pp. 1-7.

[28] Bullet-revolutionary outdoor radio device, "'http://www.ubnt.com/bullet,".

[29] 3-dBi omni-directional antenna, "'http://www.l-com.com/wireless-antenna-24-ghz-to-58-ghz-3-dbi-triband-rubber-duck-antenna-sma-male,".

[30] Chris Veness, "'http://www.movable-type.co.uk/scripts/latlong.html,".

[31] T. Goff, N. Abu-Ghazaleh, D. Phatak and R. Kahvecioglu, "'Preemptive routing in ad hoc networks," *Journal of Parallel and Distributed Computing*, vol. 63, no. 2, pp. 123-140.