DISTRIBUTED SENSING WITH FAULT-TOLERANT RESOURCE REALLOCATION FOR DISASTER AREA ASSESSMENT

By

Adrian P. Lauf

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

May 2010

Nashville, Tennessee

Approved:

Professor William H. Robinson

Professor Gabor Karsai

Professor Alan Peters

Professor Mitch Wilkes

Professor Yuan Xue

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS

This work, and the long, yet fast-paced road through graduate school in preparation for this event was in a large part due to the incessant efforts and guidance of Dr. William H. Robinson. His style of letting me make my own experiences, both good and bad, has provided me with all the tools that I needed to bring myself and my work forward to this point. With endless patience, never once discouraging me, even in the face of a poorly-compiled draft – or three – he has always supported me in all my efforts to pursue my graduate school career from the moment I started to work with him. For all his efforts, and for the preparation he has given me, I will always be grateful.

I also have received much help and support from my Dissertation Committee, Drs. Gabor Karsai, Alan Peters, Mitch Wilkes, and Yuan Xue, who have stood by me during my progress, helping me to refine my ideas from theoretical concepts to full-featured implementations. I thank them for helping me to explore my research space, to understand what I was trying to accomplish, and for always being available when I had questions. Lastly, I thank them for providing me with an atmosphere that was always supportive, providing me with the confidence that I needed to surmount each obstacle.

My ultimate debt of gratitude goes to my parents, Dr. Peter K. Lauf and Dr. Norma Adragna-Lauf, for providing me with tireless love and support, through good times and bad, that has helped lead me to become the person I am today. With their constant backing and care, I was provided with all the emotional and environmental support that I needed to flourish in graduate school. The culmination of this work is in part a testament to their efforts.

This work was supported by TRUST (The Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: AFOSR (#FA9550-06-1-0244) Cisco, British Telecom, ESCHER, HP, IBM, iCAST, Intel, Microsoft, ORNL, Pirelli, Qualcomm, Sun, Symantec, Telecom Italia and United Technologies. Additional airframes (Telemaster 40) were supplied by Gene Kerr, member of the Edwin Warner Model Aviators club in Nashville, Tennessee.

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ADS-B ..... Automatic Dependent Surveillance – Broadcast

AP ..... Autopilot

AODV ..... Ad-hoc On-demand Distance Vector Routing

AES ..... Advanced Encryption System

CG ..... Center of Gravity

CISC ..... Complex Instruction Set Computer

CPU ..... Central Processing Unit

DARTS ..... Distributed Apt Resource Transference System

DARTBOARD .... - Distributed Apt Resource Transference for Broad-response Overhead Airborne Reconnaissance Dispatch

DSSS ..... Direct Sequence Spread Spectrum

ESC ..... Electronic Speed Controller

FAA ..... Federal Aviation Administration

FOSS ..... Free Open-Source Software

GPS ..... Global Positioning System

IDS ..... Intrusion Detection System

GCS ..... Ground Control System

GSD ..... Group-based Service Discovery

IPv4/IPv6 ..... Internet Protocol version 4 (or version 6)

MANET ..... Mobile Ad Hoc Network

MEMS ..... Micro Electro-Mechanical Switches

MER ..... Mars Exploration Rover

MITM ..... Man-in-the-Middle

NOAA ..... The National Oceanic and Atmospheric Administration

OLSR ..... Optimized Link State Routing Protocol

PSO ..... Particle Swarm Optimization

RAT ..... Rock Abrasion Tool

R/C..... Radio Control

RISC ..... Reduced Instruction Set Computer

RREQ..... Route Request

RREP ..... Route Reply

TCAS ..... Traffic alert and Collision Avoidance System

TCP/IP ..... Transmission Control Protocol/Internet Protocol

TDP..... Thermal Design Power

TIP ..... Transference Inquiry Packet

UAV..... Unmanned Aerial Vehicle

WAN..... Wide Area Network

CHAPTER I

INTRODUCTION

When considering technologies that pervade society and daily life, few come close to the sophistication and permeation of embedded systems. These special-purpose, applied computers provide services for a wide range of systems, such as transportation control, avionics, chemical plant control, communication, and even personal electronics [1-3]. These systems are so popular, that over 99 percent of all microprocessor integrated circuits in production as of the time of this writing are intended for use in embedded systems [4]. An added benefit to embedded systems is that workloads spanning large and/or complicated services can be distributed and partitioned to a number of strategically-placed sub-systems (i.e., agents) by using networking technology. This enables an improvement in system monitoring and control; the distributed agents can increase productivity and functionality of the overall system.

As distributed systems increase, so do the challenges of operating them in a reliable and safe manner. In addition to reliability concerns, distributed and networked embedded systems face natural impediments to their operation, such as radio-frequency interference with their wireless communication systems, or malicious attacks designed to compromise the operation of a system. Both of these types of interference threaten timely system operation; in cases where hard real-time deadlines are critical to both task execution and the overall goals of the system, such interference could lead to devastating consequences [2, 5-7]. To address that threat, the research work presented in this dissertation aims to provide a real-time, lightweight solution to allow distributed systems

to identify interference vulnerabilities, assess performance degradation, and circumvent security breaches. A specific application, namely disaster area assessment, is highlighted to demonstrate the feasibility of the technologies and methods developed during this dissertation. The author believes that unique life-saving benefits can be provided by integrating fault-tolerant distributed sensing mechanisms with disaster area assessment methods. These can be used to identify damage on the ground following localized weather phenomena such as tornadoes, microbursts, and straight-line winds. In particular, the dissertation will focus on fault-tolerance within the scope of mobile ad-hoc networks (MANETs) [8, 9].

The design of fault-tolerant distributed systems involves the understanding of basic network architecture, and how its features can make it vulnerable to natural or man-made forms of interference [10, 11]. As such, certain challenges must be addressed in order to develop a robust solution that meets the requirements set out in this dissertation. The first challenge is to identify a candidate network environment that would benefit from a specific fault-tolerant approach. The second challenge is then to understand what aspects of the network are most at risk. These include weak encryption schemes, non-robust networking architectures (for instance, the lack of direct-sequence spread spectrum radio technologies), and single points of trust and responsibility. Because this dissertation will focus on MANETs in particular, the author is restricting his scope to mobile networks implementing a form of IEEE-standardized networking technologies, such as 802.11x (WiFi), 802.15.1 (Bluetooth), and 802.15.4 (ZigBee) [10, 12]. This restriction lessens the focus on implementation details of the networking infrastructure, and instead enables the examination of system behaviors, application profiles, node proximity, and

node mobility. A proposed solution must also meet the challenge to provide an acceptable improvement over the state-of-the-art.

The research described here presents a solution to the unique challenge of maintaining fault-tolerance in a mobile network scenario. By improving upon two key mechanisms of discovering available resources on a network with homogeneous or semi-homogeneous devices, reallocation can be performed faster and more efficiently; this improved reallocation results in a system that is more stable and has a higher effective uptime, especially when coupled with intrusion detection system (IDS) technologies, as shown in this work. This dissertation describes three specific contributions:

1.) developing a method to reallocate resources on a compromised mobile ad-hoc network, assuming the network is equipped with redundant resources to provide fault-tolerance,

2.) analyzing the impact on performance of this proposed technique on networks of variable size and connectivity, and

3.) applying the findings of this method toward a real-world application – aerial disaster area assessment using multiple, distributed unmanned aerial vehicles (UAVs.)

4.) During the course of researching and developing this topic, a number of challenges manifested themselves, including the need to understand the correlations among various network topologies, the density of the networked nodes, and mobility. These challenges required work to clarify how they affected system performance. Performance is measured by determining the time required

to find new resources, as well as by counting the number of messages transmitted on the network to find new resources. The work presented in this dissertation yields a baseline performance gains of 6-fold speedups, and a required message count reduction of up to 80% from end to end. The speedup and message reduction was analyzed in the context of differing network structures. The benefits of the proposed system, which is called DARTS (Distributed Apt Resource Transference System), was then integrated into the final implementation involving aircraft-based distributed sensing for disaster area assessment, called DARTBOARD (Distributed Apt Resource Transference for Broad-response Overhead Airborne Reconnaissance Dispatch). Thus, the challenges for developing a reallocation method that: (1) improved system uptime, (2) reduced the overhead of reallocating a resource, and (3) applied this knowledge into a workable real-world system for assessing storm damage, have been met and outlined in this dissertation.

5.) The dissertation will be organized into nine chapters. Chapter II introduces mobile ad-hoc networks and their characteristics. Chapter III introduces reliability and security concerns, such as identification of specific threats, including jamming, spoofing, man-in-the-middle attacks, and similar intrusion strategies. Chapter IV explains the basic principles behind tasks, task reallocation, resources, and the proposed method. Chapter V details the method's performance on variably-sized networks. Chapters VI through VIII combine for the scenario of disaster area assessment using UAVs. Chapter VI details weather-related phenomena, namely tornadoes. Chapter VII covers material related to aircraft and

configuration for the implementation seen in Chapter VIII. Chapter VIII seeks to outline the implementation of the fault-tolerant method in a real-world example, using actual in-field data and network simulation cases. Lastly, Chapter IX discusses conclusions and future possibilities for the work outlined in this dissertation.

CHAPTER II

MOBILE AD-HOC NETWORKS (MANETs)

This chapter covers the essentials of ad-hoc networking, and their associated networked nodes, which consist of embedded systems. System design constraints, such as power and size limitations are introduced to aid the reader in understanding why certain design choices have been made in the implementations detailed in later chapters.

By definition, a mobile ad-hoc network, or MANET, is a subclass of a basic ad-hoc network. This dissertation defines an ad-hoc network as any loosely-defined network between two or more nodes (e.g., embedded devices) such that nodes may enter, be acknowledged by the group (i.e., node discovery), or leave based on pre-established protocols of discovery [8, 9, 13-15]. Nodes may be fully or partially connected – meaning that clustered groups may lie outside the range of communication of another group [16]. Therefore, we can extend this definition to include MANETs such that the networked nodes include at least one or more nodes that are mobile. MANETs are relevant to our discussion of distributed systems because a typical multi-agent distributed sensing implementation relies on networked nodes that fit the definition of an ad-hoc network, and furthermore have a tendency for mobility [9]. The reasoning behind this will be discussed in significant detail in Chapter V.

Networking

Networking is the fabric that binds together ad-hoc network nodes. For static systems, range and expected environmental interference are factored into the network's design and protocol selection. Because ad-hoc nodes are designed to enter and leave

based on requirements, power supply, or mobility situations, wireless networking protocols are typically preferred. A subset of wireless networks, such as the 802.15.4 ZigBee protocol are used because they are optimized to work with a decentralized network architecture [12]. Mobile systems are more difficult to design, since range is generally not constant, thus making a hierarchal network infrastructure difficult to implement [12]. Let us briefly examine the difference between hierarchal and distributed (i.e., ad-hoc) network instances.

Principles of Hierarchical Networks

Hierarchal or *infrastructure*-based networks assume a central point of connectivity, which is connected to a routing system. All traffic must go through the main access point (AP, the antenna-like cone in Figure 1) before being directed to other nodes connected to the same or other access point. The advantage to this system is that routing and client models are well-defined Also, connectivity is simplified if the system is provided with a standardized binding protocol, such as found in 802.11x, and a standard transport protocol, such as IPv4 and IPv6 [16-18] bandwidth must be shared and divided among nodes. Figure 1 shows such a sample of such a centralized system. However, there are some key disadvantages for general network use, and other disadvantages that make infrastructure-based networks unfeasible to implement for mobile systems. The first is that bandwidth must be shared and divided

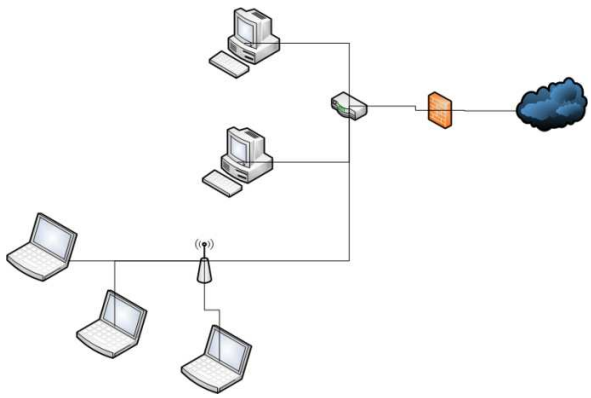

Figure 1 - A traditional centralized network - the router is connected directly to the firewall

among nodes because all wireless nodes must connect to a localized access point. This can cause resource competition problems, and can threaten the operation of bandwidth-critical applications should too many nodes attempt to join the access point. Second, an infrastructure-based network is not often made of mobile components, especially when considering the access point itself. However, mobile wide area network (WAN) routers exist and are becoming popular for use in mobile computing applications, allowing multiple client computers to connect to internet resources wirelessly. Despite a trend towards mobility, centralized networks cannot account easily for range issues in a mobile, distributed system. For instance, an access point may be able to cover several nodes, but not others that have moved out of range. More access points, range extenders, and repeaters can be installed, but at considerable expense and complexity when compared to an ad-hoc network.

Principles of Ad-hoc Networks

In contrast, distributed, or *ad-hoc* networks are based on device-to-device mesh networking. [19] Seen in Figure 2, the device nodes form links to nearest neighbors. Compared to infrastructure-based networks, this offers several advantages; network links can easily be established and can accommodate mobility by allowing dynamic link establishment. Furthermore, if a link is degraded, then other routes may be available, allowing communication to bypass an unresponsive device. This bypassing method, however, is determined by topology – a network comprised of serial nodes could easily be threatened by a single link failure. Of course, disadvantages exist as well;

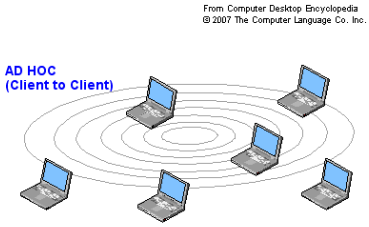

Figure 2 - An ad-hoc network example

routing is considerably more complicated. The process of routing allows messages to be sent from a source to a destination, using intermediary nodes or networking equipment [19-22]. With infrastructure-based networks, such as Ethernet and WiFi networks, the static nature of the network setup generally makes routing simple – a computer will likely remain at a particular port and address mapping for the duration of the needed transactions. This can be observed when a computer switches access points while attempting to maintain connectivity; any open sessions drop and must be re-established. As a result, this means that infrastructure networks do not support roaming. In ad-hoc networks, routing must be determined either on-demand, or by using a polled or gossip-based update method, allowing nodes to understand how traffic should be routed. Both on-demand and active routing discovery methods are significantly more complex than those found in infrastructure networks. Such methods include reactive methods like Ad-hoc On-demand Distance Vector routing (AODV) [23], which focuses on finding optimal routes in highly dynamic systems, as well as table-based routing protocols such as the Ad-hoc Wireless Distribution Service (AWDS) [24] and the Destination-Sequenced Distance Vector Routing (DSDV) [25], which utilize routing tables between nearest-neighbor nodes that are periodically refreshed as needed. Regardless of the method used, routing is central to effective data distribution. It represents a fundamental transmission method, and is second in importance only to physical connectivity such as wireless radios and Ethernet lines when considering MANET connectivity. Because routing is so critical, it also represents a choice point of attack, as disrupted routes can easily destroy any network coherency almost instantly. If data cannot reach its intended target, any tasks dependent on multiple nodes may cease to execute completely, causing service disruption

or even physical damage to the overall system that depends on the MANET. Based on this threat model, much work has been done to identify security intrusion breaches by utilizing Intrusion Detection Systems (IDS) that map traffic patterns and correlate them based on MANET routing performance [23, 26-41]. This dissertation does not focus on routing-based IDS techniques. The intrusion detection methods applied by this research will be discussed in Chapter III.

Lastly, ad-hoc networking faces the need for range management, as well as roaming capabilities. Under ideal conditions, network connections are retained or suspended if a node moves out of transmission range and back. Most ad-hoc network protocols are now being implemented using the 802.15.4, or ZigBee protocol. Flexible, power-conscious, and reliable, 802.15.4 offers a good compromise of bandwidth and transmission range [10]. Only ad-hoc, wireless networking will be discussed in solutions and implementations during this dissertation.

Routing depends on the availability of a physical network layer through which packets may be sent and received. For most MANETs, this is accomplished using a wireless radio stack that corresponds to a variety of wireless protocols. Such protocols include: (1) the now-ubiquitous 802.11 wireless transmission protocol, or WiFi [11, 16-18], (2) a low-power, low-throughput protocol known as 802.15.4 [10] and its derivatives, and (3) 802.15.1-2002, known more widely as Bluetooth [10]. Others still employ digital cellular data networks, such as 1xRTT and Ev-DO for the Code Division Multiple Access (CDMA) network standards, and UMTS, HSDPA/UPA, and GPRS for the Global Standards for Mobile (GSM) standard [42]. In each case, the radio stack provides a data transceiver which best fits the target application.

802.11 and its derivative implementation sub-standards (i.e., a, b, g, and draft "N"), provide a high communications throughput by using multiple send/receive antennas and channels, reaching up to a theoretical 300 Mbit/s in the case of draft "N" [11, 16-18]. A logical downside is the large power requirements of the radios for transmitting data at such rates. Thus, a solution for networks requiring less throughput but needing power conservation can be found in the 802.15.4 protocol, which allows for a reasonable range of connectivity and low power requirements, at the expense of throughput [11, 16-18]. For the duration of this work, it is assumed that network-layer connectivity has been established through one of the protocols mentioned above. There will be no significant further discussion on the physical network layer.

MANET Node Discovery

Another key shared components between most if not all MANETs is that of a discovery protocol [8, 9, 13, 15]. Discovery allows nodes to identify new, incoming devices that wish to join a MANET. Discovery protocols work in one of two principal ways: *push* and *pull* [8]. Push-based discovery protocols instruct existing nodes to broadcast information about the MANET to any incoming new nodes. Pull-based discovery protocols allow the MANET to respond to broadcast requests from incoming nodes. There exist many protocols that support both push and pull mode of node and/or service discovery.

Based on security, performance, scalability, and power requirements of the MANET, there will likely exist a unique protocol that can satisfy most of the individual requirements. For instance, a MANET that requires extremely low power consumption

with some method of security might use a public key exchange security protocol in combination with a pull-only discovery method.

Discovery also provides a window for attack, by which a malicious node or agent may choose to obscure the discovery process by jamming, by falsifying handshakes (i.e., a mutual identification establishment that occurs during the discovery process), by spoofing other nodes, or by reporting incorrect information about the MANET to which a node wishes to join, thereby causing it to join a network that may not be the intended target. The security threats will be discussed in detail in Chapter III.

A corollary to the discovery process could be called pruning, which is the excision of nodes by the MANET, or their voluntary (or involuntary) removal by their own operational processes. Pruning may occur for a number of reasons; a MANET may be at full operating capacity and at risk of network communication problems such as data packet collisions or too much crosstalk over the wireless frequencies. Or, a node may simply be passing out of the range of the network, such as an aircraft continuing on its journey after forming a brief network with surrounding aircraft. Such an example can be seen in the Automatic Dependent Surveillance - Broadcast system, or ADS-B [43-46].

Embedded Systems Properties

Let us now shift our focus to understanding the basic components present in a MANET, and then discuss how these components operate interdependently in a networked situation. Our first point of discussion is in regards to the embedded device node itself. Because embedded computers are designed (both from hardware and software perspectives) to operate within a limited range of applications, care must be taken to

observe four significant limitations in the operational capacity of an embedded device node. These limitations and characteristics are, according to implementation difficulty experienced during this research work, are in order of design importance: (1) power supply limitations, (2) processing power limitations, (3) weight and size (physical) limitations, and (4) in our mobile case, the mobility concerns themselves. Let us briefly analyze these four points to better understand hardware/software co-design issues.

Power Supply Limitations

The power source of a MANET node can be variably restrictive. This is true in the sense that a device may be powered in different modes at different times, in addition to the fact that some systems are designed with limited, but portable and independent power sources such as batteries and photovoltaic cells. Others have significant power reserves such as direct line power, or portable generators with long fuel supplies. In the case of a restricted power supply, such as a battery, the node must be designed with tradeoffs of computing power and energy consumption rates in mind. As a side-note, lower power consumption generally equates to lower thermal dissipation, meaning that the CPU requires less cooling, a weight advantage in system design [47, 48]. Power source restrictions may come from a variety of different sources. In mobile systems, where weight and size constraints may be key (e.g., in the implementation of ultra-micro-scale aerial vehicles), batteries must be designed around the physical characteristics of their intended chassis, as well as conform to maximum weight guidelines so as not to cause problems with mobility concerns. Other conditions may arise from the environment. For instance, lithium-ion-polymer batteries, though lightweight, moldable, and containing a high energy density, are still subject to efficiency concerns when

operating at low temperatures; the organic solvent to which the lithium ions are bound is ultimately a liquid [37, 49]. Thus, a device operating on such a battery could experience lower projected runtimes because of its location – a concern that needs to be addressed in system design stages.

Processing Power Limitations

The processing capabilities of embedded systems are *typically* dictated by power requirements, and are a direct consequence of power supply constraints; a palmtop computer or Smartphone running a quad-core Intel Core i7 processor (industry-standard, high-performance microprocessor as of the time of this writing) would likely exhaust its battery supply within minutes, as such central processing units (CPUs) can draw upwards of 135-150 watts, depending on their thermal design power (TDP) [50]. Not only would the power source be drained quickly, but the system would then require a massive cooling solution that might weigh more than the device itself; modern computers as of the time of this writing often employ large copper heat sinks and fan units that together can weigh over 1 kg.

This is another reason why mobile processors, typically requiring less-complex instruction sets, and fewer on-chip performance features, are typically RISC processors, which due to their smaller instruction set have a lower transistor count, drawing less power, and needing only passive cooling (in which a fan or liquid dissipation system is not implemented) through the system's case, or in rare cases, a small aluminum heat sink. Home internet routers, smart phones, unmanned aerial vehicle control systems, and many other embedded system types typically use processors that conform to the ARM 7 or ARM 9 family of RISC processors [51]. In this dissertation work, the autopilot system

implemented in a candidate remote distributed sensing aircraft, utilizes an ARM7-compliant microcontroller platform. The 32-bit ARM-based microprocessors and microcontrollers are therefore flexible and reasonably powerful for their category, and can be clocked and modified (such as adding multiple execution superscalar architecture, as seen in the Cortex A8 variant of the ARM11) to improve performance, and put in multi-core arrangements (such as the Cortex A9, a dual-core, superscalar architecture variant of the ARM11 [51].) This capability, in combination with enhanced manufacture technologies that permit smaller feature size, can create a usable tradeoff between performance and power utilization. ARM-based RISC processors are also generation-backwards-compatible, meaning that they are able to execute code compiled for previous intellectual property generations. The ARM processor model is not a specific piece of hardware, but rather an intellectual property library licensed to chip manufacturers for their specific implementation. An example of this is found in mobile phones: a CPU manufactured by Qualcomm called the Snapdragon is instruction-set compatible and derived from the same ARMv7 (not to be confused with the ARM7) architecture as a chip manufactured by Texas Instruments called the OMAP3440 [52].

Weight and Size Limitations

Size constraints dictate the footprint of the embedded system. In mobile cases, this is especially important, as oversized components such as sensors, processing electronics, and radio communications equipment, and associated power sources such as batteries, can cause significant challenges for mobility. In the case of autonomous ground vehicles, for instance, mobility on a sloped path might be compromised by a battery that is too heavy of a load for motors to counteract. Further complications could arise if the

vehicle were dependent on wheels, where added weight could compromise traction in more difficult terrain. Internal mechanics may also face design challenges if control systems are too large to permit free operation of belts, shafts, gears, and other components.

Mobility Constraints

The last major characteristic of all MANETs is derived from its name: mobility. Mobility adds an increased requirement for maintaining network cohesion, as nodes may be drifting in and out of communication range at all times. This stresses a number of key components; for instance, a node that is barely within range may cause wireless transmission rates to drop drastically in order to maintain communications with an increased presence of noise. The change in data rates can, in turn, cause a failure of dependent tasks on the MANET, since the task may no longer be able to execute in real time with slower peer connectivity. One consequence is the increased requirement for radio transmission power necessary to maintain communications cohesion; this is a problem for any system reliant on a limited power source.

The MANET as a whole must also face variable operating conditions. If the system were to stray into a boundary with increased radio interference, then the entire group may fail catastrophically without a chance for recovery because sufficient nodes are unable to compensate for the interference. Or, conditions may change that prevent the operation of one or more critical nodes needed to carry out a task. Whatever the case may be, MANET communications protocols, task allocation mechanisms, and distribution mechanisms must be ready to overcome limitations of the environment in which they

must operate, as well as possible unknown obstacles that may interfere in unpredictable ways with their general operations.

CHAPTER III

CRYPTOGRAPHY AND DETECTION

This chapter covers essentials on computer security, and its applications towards embedded systems. Topics on attack methods and fault conditions are introduced to foster an understanding on some of the challenges that embedded MANET nodes face. A three-level hierarchy for comprehensive security is introduced following classification of the threat model.

Security and Reliability

Attacks and node failures motivate the creation of the intrusion response mechanism that will be detailed in this work. Within the context of a MANET, an attack constitutes an intentional disabling or disruption of software, services, or hardware of one or more nodes on a network. This causes a degradation or cessation of the MANET's functionality. An attack challenges some or all of the five basic tenets of network security: (1) data integrity, (2) confidentiality, (3) availability, (4) authenticity, and (5) non-repudiation [53].

An attack on data integrity consists of modifying the contents of data during transmission over the network, or while resident on a device node. For instance, in the communications between two aircraft over a channel, a third party may attempt to maliciously alter data being sent from one aircraft to another so that the nodes might incorrectly identify their relative positions. To combat attacks on data integrity and other core security aspects, there exist a number of passive intrusion prevention techniques

called cryptographic services. Crypto services will be explained in more detail later in Chapter III.

Confidentiality attacks seek to identify or disclose information in transmission along a network. Returning to the networked aircraft example, suppose that vital information on either a target or destination for an operation is being transmitted between devices. This information could be valuable to a third party in conflict with the operation being performed. The confidentiality attack would aim to discover the contents of the transmissions to thwart the objectives of the network. While it may seem that an integrity attack must be a subset of a confidentiality attack, this is not necessarily the case; weakly-encrypted data packets can be modified without discovering their contents; however, confidentiality attacks are generally precursors to integrity attacks [53].

Availability attacks are one of the easiest types to implement. Generally, an availability attack is designed to prevent the flow of information across a network. In this case, one or more nodes attempting to transmit data are unable to do so for an undetermined period of time, which can cause a partial or complete disruption of services on the network. These typically fall under the Denial of Service, or DOS category. On a wired network, availability attacks can be implemented as a flooding of the hierarchical networking components such that network packets can no longer be routed successfully from source to destination. On a wireless network, jamming the wireless carrier frequency can achieve the same result, though with varying degrees of effectiveness based on the location(s) of the jamming device(s).

Authenticity is the guarantee that the source and destination of a transmission can be validated as genuine. In particular, authenticity ensures that no unwanted third party can be the source of a transmission. Attacks on authenticity seek to undermine the capability of a network to accurately identify the source of a transmission, as well as the identity of the receiving node(s). Because an authenticity attack is difficult to perpetrate and requires intimate knowledge of network protocols and node identifications, this dissertation will spend less time focusing on this type of attack.

Lastly, non-repudiation seeks to ensure that a receiving party cannot deny the receipt of a transmission, and that the originating sender cannot deny having sent it. This property is useful in trusted communications where unique data payloads may be transmitted sequentially or in a time-critical manner. This dissertation will not focus on non-repudiation attacks.

The attacks on the various network security principles can be accomplished through service denial (Denial of Service or DOS attack) and jamming (a type of service denial), impersonation (Spoofing), eavesdropping (Sniffing), data modification, and man-in-the-middle attacks, which can accomplish any of the above [53]. Each of these techniques involves differing levels of implementation difficulty, network access, and target unique portions of the network.

Service Denial via Jamming

As mentioned earlier, a jamming attack consists of flooding or overwhelming the network transmission capacity of one or more networked links. Typically referred to as a Denial of Service attack, jamming attacks are simple to implement because they require

no trust or authentication from their targets. Using the networked aircraft model as an example, a jamming attack would manifest itself as a radio transmission on a similar frequency spectrum as that used to communicate between the aircraft nodes. At sufficient strength, this rogue signal is capable of overwhelming the receiving radio transceivers on the nodes by lowering the signal-to-noise ratio to unacceptable levels. At these levels, data cannot be distinguished from the noise, and the transmission is lost. The jamming device needs no privileged network access – or any network access at all. Armed simply with the knowledge of transmission frequency spectrum, the attacker can disable one or more nodes depending on its intent, proximity, and the number of jamming devices used. The properties of a jamming attack – its overpowering of a network interface, and the apparent localization of the jamming source on a wireless network, make it one of the easiest to detect. A network suspecting a jamming attack will notice remarkably reduced data transmission rates across communications links that seem to be localized. The source of the jamming signal can then be identified.

When considering the five aspects of network security, jamming affects availability. It also indirectly affects non-repudiation, and loosely, data integrity – though at this point no data is received. Similar analogs exist on wired networks, in which the hierarchical routing structure is bombarded with network traffic (such as a SYN flood, consisting of TCP/IP packets that initiate a connection request but never completely establish it – flooding the networking device with useless requests). The intended target is no longer able to route information at a reliable rate. In addition to the inability to route and send data, jamming has an indirect effect on performance: it markedly reduces battery life of devices experiencing the attack; if a communications link is severed,

multiple retries along the channel will be necessary. In addition, increasing power to the radio transmitter modules may be required, further reducing available power.

Spoofing Attacks

Spoofing is the behavioral dissimulation of a trusted node by a malicious attacker node. The purpose of a spoofing attack is to damage a node or to inhibit a task's execution by maliciously issuing commands that appear to originate from another legitimate node. In the context of the five principle of network security, spoofing affects data integrity (i.e., a spoofing node that serves as a routing device may, for instance, re-write data that is meant to be transmitted along to another node), data confidentiality (i.e., spoofing can lead to eavesdropping), authenticity (i.e., the spoofing node is trusted as if it were someone else), and to some degree, availability, (i.e., the spoofing node can potentially stop the routing of information as if it were a normal occurrence). Because it affects so many aspects of network security, spoofing is regarded as a dangerous attack because it gains the trust of the nodes on the network. This allows the attacker to inflict nearly limitless damage on the network without any means of discovery by conventional, passive, cryptographic, intrusion prevention policies. Consequently, it is also one of the most difficult attacks to perpetrate, since a well-designed network will feature several layers of protection that must be individually broken before spoofing is possible.

Referring once more to our networked aircraft example, a spoofed node could misdirect information about the position, speed, and objectives of other aircraft on the network, deliberately causing a collision or failure of the joint objective. In contrast to the jamming attack, which is more random in nature (even though a particular node may be

targeted, the effect of cutting off communications may not be immediately known to the attacker), a spoofing attack has direct and well-defined consequences, as the attacker is responsible for causing changes at the very fundamental level of the network. In addition, using intrusion detection technology at this level would require significant knowledge of the application and its behavior in order to efficiently detect the presence of an attacker in a timely manner. To further obfuscate its presence, a spoofing attacker may choose to behave in a manner that seems sufficiently normal so as not to raise suspicion, particularly from an anomaly-based intrusion detection system.

Spoofing can most easily be implemented during the node discovery phase. Should the attacker, represented by its own node, properly craft its discovery credentials, the node may be convinced that the attacking node is in fact another device, and thereafter assume it to be a trusted device without further investigation. An attacker may also be able to compromise an existing node on the network that has properly established trust. In this case, the node may be reprogrammed or altered in such a way that it still retains its trusted relationship, but now broadcasts information generated with malicious intent.

Compromising Confidentiality with Eavesdropping

Eavesdropping involves capturing and retransmitting packets (or, alternately, receiving copies of a network packet by a node that is not intended to receive it) to identify its contents. These contents might be passwords, objectives, or any other sensitive data that should not be discovered by non-trusted parties. This is pertinent to the data confidentiality aspect of network security. On wired networks, eavesdropping is

typically performed by using a packet-sniffing software application that allows the system's network adapter to pay attention to packets for which the adapter is not a designated recipient. The adapter is said to be operating in promiscuous mode. During an eavesdropping attack on a wired network, unencrypted network packets are the easiest target. However, today's networks more frequently make use of encryption and authentication methods, such as Secure Shell (SSH) (application-layer), or Internet Protocol Secure (IPSEC) (internet layer, between transport and link-layer on the TCP/IP stack) to prevent sniffing from identifying the packet contents. Still, depending on the cryptographic method in use, it may still possible to decrypt packet contents using known plaintext attacks, for example, by exploiting a weakness in the encryption method.

On wireless networks, particularly MANETs, eavesdropping may occur through reception of network data transmissions either through a node that is associated with the network, or through brute-force recording of all radiofrequency spectrum activity. The former requires a trusted relationship with the network, assuming that the nodes are operating on a secured communications system. As mentioned earlier, a spoofing node may be party to an eavesdropping attack. The latter is much more difficult to detect on a wireless network, if at all possible. Thus, intrusion detection technologies are rendered useless on eavesdropping attacks since there may be no known signature or anomalous presence on the network that can be used to identify an attacking device. The only defense, in this case, is a robust encryption method to ensure data confidentiality.

Man-in-the-middle Attacks

A man-in-the-middle (MITM) attack occurs when a third party receives or retransmits a portion of the data in a modified manner. During the attack, the two originally-communicating parties do not perceive the existence of the third party, or instead see it as a trusted resource [27, 53]. This attack can be used for eavesdropping and for data modification. Data modification is as simple as it sounds – a violation of data integrity by means of retransmitting or altering network packets and their contents to reflect new content with malicious intent. The man-in-the-middle attack faces implementation challenges that are similar to a spoofing attack. In the case of authenticated, encrypted network connections, the attacker must intercept an authentication token, such as a public key, in order to execute the attack. When implemented on a wireless ad hoc network, the man-in-the-middle attack essentially behaves in the same manner as a spoofing node.

In a wired, infrastructure network, public keys used in cryptography can be issued from a certificate authority and verified such that the intercepted key cannot be used without verification (discussed later in Chapter III). On an ad hoc network, however, such a system may not exist. Detection of a man-in-the-middle attack through IDS techniques can be performed by identifying latencies in transmission that are not typical of the connected nodes. Thus, an MITM attack would likely be detected by means of a signature-based intrusion detection system that can detect common symptoms of the attack itself.

Node failures without malicious intent

In addition to attacks representing malicious intent, MANETs, and more broadly almost all other ad hoc network types, are vulnerable to node failures caused by environment, hardware defects, or software programming errors [54]. Of these, the most complex and difficult to analyze and predict is the environmental factor. Environment encompasses many variables that can cause one or more nodes to fail to respond. Temperature, which can affect battery life in either extreme of hot or cold, can also cause hardware failures if the nodes are unable to properly cool themselves through either active or passive cooling elements. Because most embedded devices use low-power RISC platforms such as the ARM 7, ARM 9, ARM 11, or similar derivative instruction set architectures, [2, 51] relatively low heat is generated, and thus passive cooling is the most common method used to keep systems operating within normal bounds. Passive cooling also reduces energy requirements. Natural barriers are another factor that can affect communications performance. If mobile nodes find themselves in a location interspersed with natural, dense rock formations, such as mountains, radio transmission may be curtailed or even impossible. Also, regions containing high metallic mineral content can also affect the range and integrity of communications.

Not all environmental challenges are natural; man-made radio frequency sources may interfere with MANET operation even if the effect was not intentional. This would be analogous to a jamming attack without malicious motivation. In order to minimize effects of interference on communications bands, especially from other devices competing for bandwidth on the same frequency spectrum such as cordless telephones, Bluetooth radios, and other 802.x protocol devices, most ad hoc communications

hardware employs the use of Direct Sequence Spread Spectrum technology, or DSSS [55]. DSSS-enabled radios all share a pseudo noise symbol string (PN string) which can be used to modulate the phase of the data stream to be transmitted. This is done through cross-multiplication of the original signal by the PN string. The resulting transmission appears semi-random in nature, but contains the important property of allowing a certain amount of overlap with other existing signals in the space without degrading the transmission. The received signal is decoded with the same PN string in a reversible operation to yield the original data transmission.

DSSS is utilized in code-division multiple access networks such as cellular networks (1xRTT, Ev-DO, WCDMA, UMTS, HSDPA/UPA/HSPA+ and others [56]) not only to allow for interference-free operation, but also to allow for multiple transmissions on the same frequency band to occur simultaneously. Thus, there is a natural tradeoff between signal strength, quality, and bandwidth. The 802.11 series of networking standards also employs DSSS, with the newest draft "N" standard extending the concept even further by allowing multiple send/receive antennas (multiple-in/multiple-out, or MIMO) on different channels to enhance throughput. The 802.15.4-2006 standard, which is rapidly becoming the overwhelming choice for low-power, low-bandwidth networks, also utilizes DSSS. Recent developments show that the ZigBee implementation of 802.15.4 will soon make an entry as the de-facto wireless networking standard for home automation services and devices [1, 12, 57-59].

Despite the inclusion of DSSS and other techniques used to prevent interference from reducing or preventing transmission, communications loss is still a major factor in

MANET deployment. In aerospace telecommunications, for instance, NASA is developing a Delay Tolerant Network (DTN) protocol that allows a network to operate under the assumption that at some point, a node will either exit communications range, or fail to communicate due to interference or another environmental factor [14, 16, 42, 43, 45, 46, 60-64]. DTN networks can be implemented between satellites, between mobile robots, and between the extraplanetary systems and ground stations used to monitor their activities.

In addition to environmental concerns, defects also present a challenge to reliable communications. Of the two defect categories, hardware and software, software typically comprises the most frequent cause for a communications failure that is not related to environment or attack. A software-level error may be responsible for application-level issues, or may affect a deeper level of network operations at the stack level. In either case, programming practices for building correct-by-design systems can be used to ensure that no system enters deadlock should an error occur; due to the nature of complex software systems, eliminating bugs is unfeasible. However, proper handling of exceptions, and the inclusion of failure mode operation can go a long way in preventing widespread system outages due to defects in the software.

Metrics for failures and attacks on MANET communications

Given the failures described due to attacks and environmental conditions or defects, this dissertation briefly proposes some baseline metrics to be used that can accurately quantize node failures and associated difficulties on MANETs. According to Perkins [19, 21], there are three main performance metrics associated with a

communications channel: (1) average throughput, (2) average routing overhead, and (3) power consumption. This dissertation will focus on the first and third aspects. The first metric to be considered is the loss in data rate, and consequently, average throughput in kilobits per second. This implies that the communications channel is compromised but has not failed completely. Such a scenario might occur during a jamming attack, or during environmental interference. Supposing that a node is being jammed, there are then two effects that can be observed in terms of throughput degradation. First, a localized effect can immediately degrade throughput at the site of the jamming attack or where environmental interference may be present. The second effect is overall reduction in throughput and data rate in the network due to the reduced performance of the affected nodes. This effect occurs when one considers either routing or a dataflow in which affected tasks on the nodes produce or consume resources on the network. This effect results indirectly from the localized effect, and may impact the network globally. If at any point data from the affected nodes on the network is needed at other nodes, suspended tasks and services will contribute to a greater overall loss in network throughput. Furthermore, when considering routing, jamming one node might completely terminate connections to that node, as well as bring the MANET's communications infrastructure to a halt. With these points in mind, this dissertation will refer to two metrics related to data rates. The first metric is *percent local bandwidth loss*, and refers to the loss of bandwidth at the immediate node level. The second metric will be called *percent global bandwidth loss*, which represents the overall network throughput loss occurring under the attack or node failure scenario. Global bandwidth may be calculated as an average, where global bandwidth,

$$B_g = \frac{\sum_{i=0}^{n} b_{L_i}}{n}$$

where b_L represents local bandwidth and n is the number of connections on the network. Alternately, global bandwidth may simply be expressed as the sum of the bandwidth of all the individual connections, not expressed as an average. This dissertation will use the first method. Because most of the networks investigated here will be using 802.15.4 derivatives, most of the bandwidth figures will be measured in kilobits per second. Equally important is mean bandwidth, determined by the average throughput of data on a network, calculated similarly to the global bandwidth, but substituting average bandwidth per link rather than maximum bandwidth.

The next metric used in this dissertation is a global measure of the number of nodes rendered inactive by a jamming attack or node failure. Though this may seem rather elementary, it is important to assess how network performance degrades as more than one critical node becomes unavailable. For instance, when considering routing along a sparsely-populated node, the loss of two nearby routes that represent the communications boundary between other more densely populated segments of the network could be devastating. In contrast, if only one were lost, there might be sufficient communications range for the remaining node to bridge both network sections on its own, albeit at a reduced data rate. Associated tightly with the number of nodes lost is the time-to-resource lost. During a jamming attack, there may be a delay between the start time of the attack (and its identification by an intrusion detection mechanism), and the time when the resources represented by that node become unavailable. This metric is useful in

determining how much time is available to perform resource reallocation before services across the network become disrupted. In the case of routing, the time may be limited and short. However, for cases in which a resource is required at low time intervals, there may be sufficient time to reorganize network resources using redundant nodes or resources in time to compensate for the lost node.

Lastly, when considering the actual proposed reallocation process, there are metrics which can assess the efficiency and suitability of the reallocation algorithms. These will be explored in detail in Chapter IV, and include items such as total reallocation time, number of required network transactions, and a form of quantizing the quality of replacement resources.

Attack Protection

When considering network security, there are three principal components to providing safe and secure operating conditions for a network, regardless of its architecture; these components are intrusion prevention, detection, and elimination [28, 29, 65-67]. Each of these components contains within itself a vast amount of past research, methods, schools of thought, and most importantly, tradeoffs.

Intrusion prevention is, perhaps, the most-well-studied aspect of computer security. It combines aspects of obfuscation, encryption (to assure data confidentiality and integrity), and authentication (digital signatures) [34, 40, 63, 68, 69]. The key components to intrusion prevention are typically passive methods that make it difficult for an attacker to gain access to a network and its resources. Intrusion prevention is

therefore the first line of defense in any network security structure, delaying or deterring a potential attack.

Obfuscation

Obfuscation is the most primitive method, and therefore least resilient method of intrusion prevention. It primarily involves hiding the transmission or source of data being sent across a network. For instance, when sending data across a TCP/IP network protocol, one way of obfuscating the transmitted data is to send packets across a number of different TCP connections and respective ports at random or quasi-random intervals so as to prevent easy sniffing of a single TCP port. However, this method is easily attacked by identifying port traffic, sniffing those ports, and then reassembling data based on the absolute sequence numbers encoded in the headers of each TCP packet. Though obfuscation alone is typically an ineffective method of providing data integrity and confidentiality, when used in combination with robust encryption and signature methods, it can strengthen such methods significantly.

A more well-known obfuscation method involves non-broadcast Service Set Identifiers for wireless 802.11x networks. Before standardization of secure wireless protocols, such as the semi-secure WEP, and the more robust WPA and WPA2 methods, not broadcasting the SSID was a way of preventing unauthorized access to networks by simply not advertising their presence. However, common knowledge or simple hacking can easily circumvent this, and thus non-broadcast SSID networks are rare. The usefulness of non-broadcast networks becomes much more apparent when implementing this obfuscation with encryption. A non-broadcast, encrypted network provides two

deterrents, and thus obfuscation can provide an important first layer in passive intrusion prevention methods.

Encryption

Encryption provides a measure of data confidentiality by obscuring the contents of the data. Encryption finds its roots early, as far as several thousands of years ago, with the introduction of the cipher. A cipher represents a modification of the transmitted or stored data, or plaintext, such that it cannot be read without a decryption method. The Caesar cipher is one of the most primitive encryption methods, in which the contents of the transmission medium, in that case the alphabet, were incremented by one or more units, constantly throughout the plaintext. For instance, the phrase "encryption is passive" might have been encrypted as "fodszqujpo jt qbttjwf", where each letter was incremented by one unit. It is considered a stream cipher, in which each atomic unit of data is processed as a continuous stream [70]. While it may have been effective in its time, the Caesar cipher is obviously easy to crack, whether one knows the shift interval or not.

Today, encryption pervades every aspect of communications and secure data storage. Most techniques implement block ciphers, in which a constant quantum of data is encrypted at a time. Such ciphers include AES, DES, RSA and related methods [68, 69, 71-74]. Because of the constant nature of a block cipher, many of its variants may be easily implemented in hardware. In the case of AES (Advanced Encryption Standard), commonly used in encrypted hard disks and data communications equipment, the data blocks are successively shifted, mixed, and encrypted with a key. This can be achieved with shift registers and Exclusive OR (XOR) hardware gates. Figure 3 shows a basic block diagram of how the AES cipher works. AES is also utilized in common software

encryption packages, such as TrueCrypt, which can even provide deniability (i.e., an encrypted container or volume will be indistinguishable from random contents of the disk) of the encrypted contents if remaining space on the storage volume is properly randomized.

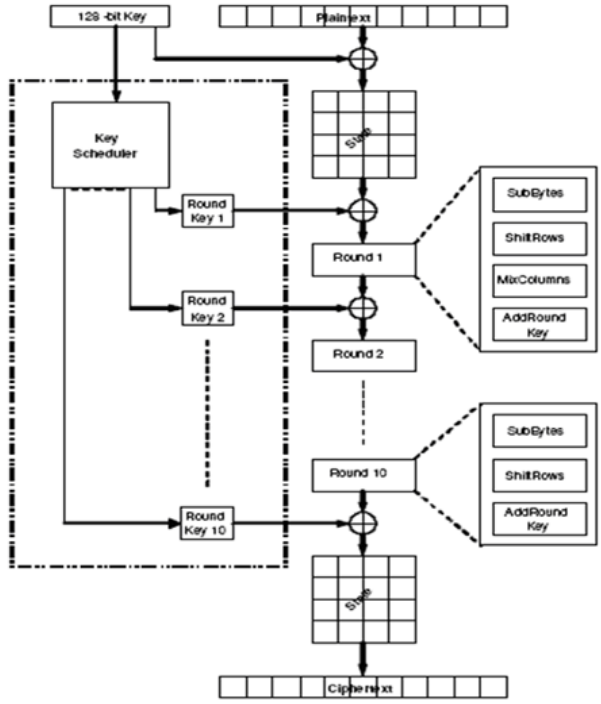

Figure 3 - AES Encryption Block Diagram

The strength of an encryption method is somewhat dependent on the number of bits used in the encryption key. If it is too short, then a "known plaintext attack" may be used to attempt to crack the encryption method. This type of attack requires some sort of generalization or assumption about the source being targeted. For instance, a PDF archive is known to begin with a particular type of header at the start of the file. If the encryption key is short enough, either brute-force cracking methods can be used (in which keys are tried in sequence or randomly until the password is cracked), or the problem may be simplified to solving a set of linear equations, because the encryption methods, with the exception of RSA (which uses exponential methods), almost always use a linear system of encryption. The known plaintext can be used as the

solution to the linear system, requiring the solution of a number of equations that generate the correct answer. In general, the method of encryption of the attack target is almost always known. Therefore, choosing an encryption key that is sufficiently large is critical in preventing either brute-force or known plaintext attacks. Because encryption relies on the presence of a decryption method, and subsequently the existence of a key or set of keys to allow encryption and decryption to take place, these methods are susceptible to attack; a strong encryption method is useless if, for instance, the key is discovered, or one system negotiates a secure connection with another system that is not who it claims to be.

Let us then consider the need for a method of authenticating sources. The most commonly used is that of a digital signature with a third party source of trust. A digital signature involves the use of a certificate that is verified by a Certificate Authority (CA). The certificate is used to digitally "sign" a public key to be used in a public key cryptography method such as RSA. Public keys are exchanged between communicating hosts, and information is encrypted with the recipient's public key. The data can only be decrypted with the host's own private key, as the public and private keys represent factorial complements. Because of the computational difficulty (at least for conventional microprocessors), factoring extremely large numbers (and thereby cracking the keys) proves to be extremely time-consuming as the size of the key increases. The importance of the CA is underscored by the man-in-the-middle attack. For example, a public key sent from Alice to Bob may be intercepted by a third party, Charlie. Charlie then resends a request encoded with Charlie’s public key (instead of Bob’s). Bob, thinking it is receiving a transmission from Alice, encodes the data with Charlie’s public key and

returns it. Charlie can then decrypt the transmission, alter it, and then re-encode it with the original public key it intercepted from Alice, sending the data back to Alice as if nothing unusual had happened. If, instead, the certificate were signed by a certificate authority, Bob would verify the contents of the public key and discover it not to be genuine. As a result, the MITM attack would be effectively thwarted. The public key's signature can be traced back to the certificate authority. Signature generation itself may use a variety of different methods, such as PGP and GPG (Pretty Good Privacy and Gnu Privacy Guard, respectively). The digital signature itself is created from a hash of the original public key's plaintext. Called a message digest, this hash will fail if the data is for any reason altered. Therefore, the public key's integrity is also protected [69, 75].

Because obfuscation and encryption methods provide passive methods of protecting data (though data may supposedly be actively obfuscated, this dissertation will assume that all obfuscation attempts are passive), any lapse in security could allow a system or network to be compromised without notice, and allow malicious activity to occur that may only manifest itself once too much damage has been done. An Intrusion Detection System (IDS) is created on the assumption that any and all passive security methods can be broken. The development and implementation of an IDS is shaped on the premise that the identification of an intruder can be exacted through monitoring a set of behaviors, conditions, or transactions occurring on a network or system [26, 28, 29, 32, 33, 36, 41].

Intrusion Detection Systems (IDS)

There are three main localizations of IDSs: Centralized, Decentralized, and Hybrid. These categories identify the *localization* of the IDS logic or scanning

mechanism. Localization depends largely on the target network architecture, throughput, available computational power, and whether or not the IDS should identify unknown anomalous behavior or a specific threat model that carries a particular manifestation signature. A large amount of research has been done on centralized, signature-based IDS methods, because they can be implemented on general-purpose, ubiquitous computing platforms [26, 29, 34, 38].

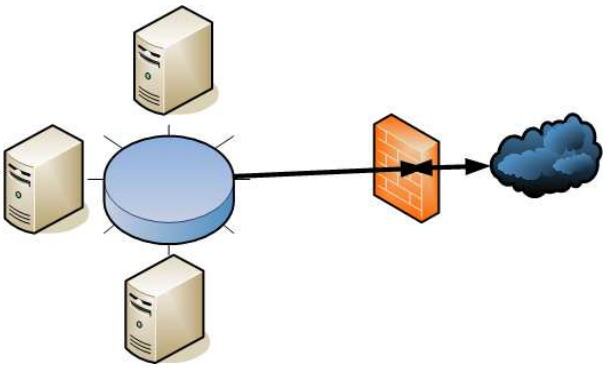

Figure 4 - A centralized IDS

A centralized IDS, as represented in Figure 4, is generally implemented at the firewall or top-level router such that it can monitor all incoming and outgoing traffic for patterns which may comprise an attack. For instance, a port scan, in which the attacking computer probes a computer or set of computers for open and responsive services that may be vulnerable to attack, can easily be detected by the range, frequency, and spread of requests made by the attacking computer. The identification can then lead to further action if desired, such as a connection termination and denial to the attacking machine. Because a centralized, routed network architecture is one of the most widely-known network topologies, IDS techniques designed to protect these systems, which include enterprise and business networks designed for general-purpose computing, have been developed for many generations and are commercially available as powerful products to protect their intended hosts [28].

On the other end of the spectrum exists the decentralized IDS. In this case, monitoring is performed at the network node level. Rather than scanning all network traffic going through the top level of a network infrastructure, a decentralized IDS performs quantized and discrete monitoring of localized events and data that may be relevant to a node and its neighbors. When paired with a decentralized network such as a MANET, this offers a number of key advantages. Primarily, it reduces the amount of computational power needed, since several if not all nodes on the MANET are performing detection roles. This eliminates the need for one entity to analyze all network packets. A decentralized IDS can also allow for a fine level of granularity in its observations. Since nodes may independently form conclusions about intrusions related to their own microcosm, a specific threat may be perceived where a generalized approach may otherwise fail to detect an anomaly. Reputation systems also fall into the category of decentralized systems. While not considered a classic intrusion detection system, the reputation mechanism, as proposed by Buchegger et al. provides a method by which nodes evaluate routing contributions based on previous activity. It is a peer-based review system, and is optimized for providing peer-reviewed routing for peer-to-peer networks [76-78].

A decentralized IDS solution also allows for application-level intrusion detection. This can remove the need for scanning packets, instead relying on application-level data to be passed to the IDS engine, implemented either in hardware or, more likely, software. Application-layer granularity allows the analysis of specific behavioral patterns that are unique to the target implementation, which may carry more meaning than network traffic which may not be indicative of an attack. For example, if in a three-node network,

containing nodes A, B, and C, let us suppose that node C has been compromised. In this network, a basic set of requests is made among all three nodes. Each request is identified by a two-digit hexadecimal code. Suppose that node C, due to an attack, begins to issue new requests to nodes A and B that are not accomplishing the network's original purpose. A centralized or packet-analyzing IDS would not notice any difference, provided that node C does not change its request issue rate. Because the packet sizes would remain the same for the two-digit request codes, any changes in application-level behavior would go undetected. On the other hand, an application-level-based IDS would notice a change in the frequency of the new event codes, and raise a flag, if it were appropriate.

A decentralized IDS also provides for fault tolerance. Should the centralized IDS become compromised in any way, the entire network it manages would be open and susceptible to attack, provided a failure of encryption and authentication methods. Even worse, it could provide a method for analyzing a target network's behavior, which could have devastating consequences. Decentralization therefore can tolerate the loss of one or more IDS nodes. Effectiveness, of course, would be reduced, but not fully eliminated. Furthermore, an analysis of failed or compromised IDS implementations can lead to improvements or identification of an attack pattern.

The third category is that of an IDS with hybrid locality. Such a system has been proposed for implementation in SCADA (Supervisory Control and Data Acquisition) network infrastructures, which are essentially MANETs connected to an infrastructure, and without the mobility component. These systems are designed to monitor and control industrial process applications, and due to the sensitive and potentially harmful effects of a network attack, IDS implementations are rapidly becoming critical to safety [57]. In a

hybrid-locality IDS, identification and control is performed on select nodes that are part of the main network. Monitoring also takes place at the central infrastructure level to which the nodes report and receive instructions. Note that most of the hybrid-locality IDS approaches focus their observation on timing principles and packet behavior, with less of an emphasis on application-layer monitoring.

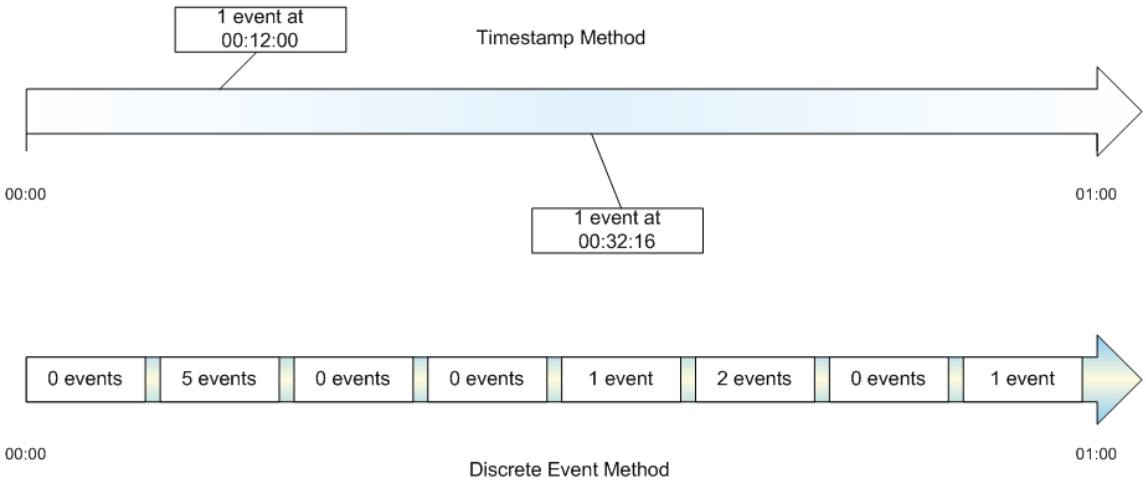

Figure 5 - Timestamp vs. Discrete Event IDS logging

IDS monitoring relies, in general, upon a set of basic components. The first component is a logging mechanism with which events can be recorded and maintained over a time period $t_0 \rightarrow t_n$. Events may be recorded according to timestamps or time steps, whichever method facilitates the desired method of analysis for the particular IDS. A particular advantage to using the time-step method is that it removes the need to account for time in computations, since all events occur as discrete event steps with each event comprising a uniform time interval. Figure 5 illustrates the two methods. The event log or cache may be flushed periodically to allow for a more accurate model of the system, in case the system has transitioned to a new mode of operation. Alternately, the event cache may be analyzed by using a sliding window, in which only the last *m* events

within $t_a \rightarrow t_n \subseteq t_0 \rightarrow t_n$ are valid. Utilizing the time-step discrete event method allows this window to remain uniform over a discrete dataset.

The actual method of analysis, sometimes referred to as the IDS engine, is what sets different approaches apart from each other. While data logging and event caches generally follow similar structures, the analysis method is tailored for a specific type of analysis and application. Some engines work based on statistical analysis of events, and determine when a particular network behavior is out of specified bounds. Such engines work well for periodic systems, such as SCADA networks in which data is designed to be transmitted and received at precise intervals. However, such monitoring can also create an enormous blind spot; an attacker can time his or her payload to execute at equally precise intervals to circumvent detection.

Other engines may analyze interactions between nodes and eschew time altogether, looking for a specific order of operations to occur. Should any interaction occur out of its designed sequence, the IDS may then raise a flag. An obvious weakness in this method is that it cannot distinguish whether or not a node acting in sequence has been compromised and is just "playing along" to go undetected. There are literally hundreds of different IDS methodologies and engines that have been proposed.

Because different attack methods manifest themselves in unique ways, no one engine will provide optimal detection efficiency for all attack types. To illustrate this, let us suppose that there exists a four-node network containing nodes A, B, C and D. Each node can communicate with the other three nodes, as seen in Figure 6. Node A controls a stepper motor to precisely align an optics array on a space-based telescope array. Node B

is responsible for measuring the temperature of the optics to ensure proper calibration, and reports its findings to node A to compensate for thermal effects. Node C receives information for targeting purposes from a supervisory communications system. Node D is redundant. Let us also suppose that node C is compromised by an attacker, with full access to the node's resources. Two identification engines exist. One detects the contents of data packets to determine if the payload of the data falls within acceptable parameters. The second engine specifically identifies the number of thermal adjustments made by node B in order to determine whether it is operating properly.

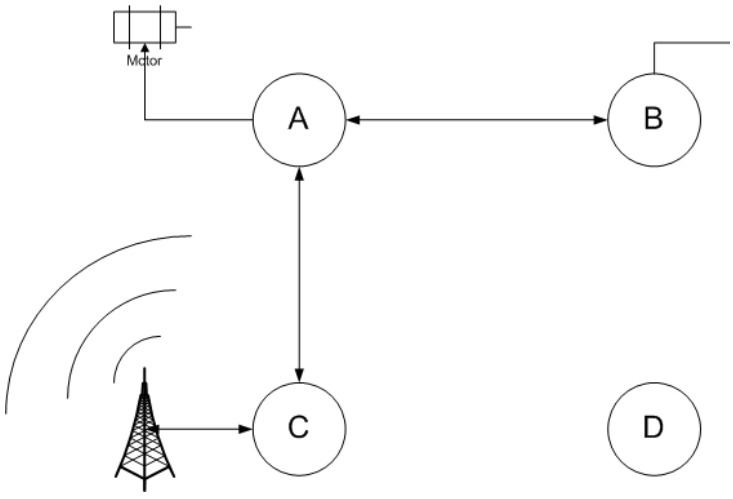

Figure 6 - Example ad-hoc network with delegated distributed tasks

Given a compromise on node C, the second IDS engine would be unable to identify any malicious behavior unless node C began interfering with B's communications link to node A. However, the first engine would likely raise a flag when the nature of the data coming from node C begins to change from a normal series of commands from the supervisory system, to commands that could potentially damage the optics array and jeopardize the overall objective. In contrast, if node B were affected, the payload and frequency might not change, but only the actual data itself, causing a misalignment of the optics and destroying the validity of

the experiment. Thus, a combination of approaches solves this elementary intrusion problem. Of course, either or both systems could be easily circumvented.

Systems that combine multiple detection approaches may be considered *fusion* or *hybrid* IDS methodologies. One such system, proposed by Lauf et al. [29], combines two engines in a time-variant IDS methodology such that one detection phase complements the operation of another. Other approaches suggest the usage of sensor data fusion and IDS engine fusion for parallel monitoring. Each engine can raise its own alert flag, and once a critical mass of alerts, or a situation-specific quorum is reached, the system is considered to be under attack from an intrusion [29].

The attack methods and the candidate MANET implementation are the two largest factors dictating IDS design. Before starting this analysis, let us briefly delve into the subject of intrusion response.

Intrusion Response

Intrusion response corresponds to the last line of defense available to a network. It is also one of the most undefined. The implemented mechanisms depend on the amount of damage an attack could cause on the network, and/or the criticality of the system under attack. For instance, in the case of a centralized IDS where all traffic must flow through a top-level infrastructure, the intrusion response of choice is to sever incoming and outbound connections from the attacking source, thereby blocking the attack entirely. Alternately, rather than blocking a connection, a MAC address, or IP address range, a particular signature of network activity may be dropped rather than routed. For example, this might correspond to port connection attempts over the NetBIOS protocol to a local

network from outside, which can occur from worms and other malware attempting to exploit improperly-secured Windows-based PC network shares. Blocking the port from all incoming NetBIOS requests solves the problem by completely disabling access to this potentially vulnerable resource, while allowing internal requests to machines over this port to continue undisturbed. Of course, this solution may have the unintended consequence of disabling legitimate requests.

Intrusion response, for the most part, is relatively straightforward for centralized, hierarchal networks. For MANETs and other ad hoc networks, intrusion response is a bit less well-defined. The intrusion response for an ad hoc network must therefore be based upon the particular network's design intent and implementation. Let us use as an example, a ten-node MANET, *C*. Each node, V_i, contains an IDS implementation. Let us suppose that a particular node, V_3, has become compromised. V_3 communicates with nodes V_2, V_6, V_7 and V_9. Since each node's IDS is capable of anomaly detection, one might envision a variety of rules that might be enacted to excise the compromised node. One rule might be that for all $V_i \in C^N$, and for all $V_j \in L^M$ (where N indicates the number of nodes, L is the locally connected group of nodes with which the compromised node is connected, and M is the number of nodes in the local group), if the number of nodes detecting an intrusion is greater than floor$\left(\frac{M}{2}\right)$, the compromised node is excised, meaning it is blocked from further communications.

Once a compromised node has been removed, the intrusion response phase has completed its goal concerning the affected nodes. At this point, it would be useful to find replacement resources on the network that represent two types of redundancy. The first

type of redundancy will be referred to as a *node clone*. In this situation, a complete copy of the replacement resource happens to exist somewhere on the MANET. This is a likely scenario for networks featuring large number of homogeneous devices, such as the massively-distributed micro robotic exploration and sensing platform proposed in [79]. In such cases, it is assumed that node failures will occur at some point during normal operation. We can extend this pretext further by assuming that one or more nodes will be *attacked* during the course of normal operation, and therefore include "hot spare" resource nodes that can be thrown into the mix at a given time. Of course, such a solution may have significant overhead concerns: extra resources, communication, and memory space may be required to maintain an understanding of these nodes, which will be using power even when they are not fulfilling a purpose. An alternative to a node clone is a *virtual node clone*, in which a node is able to divert part of its processing time or other resources to a task as if the node were exclusively assigned to that task. In reality, this builds on multiprogramming and multiprocessing capabilities and resource sharing to allow the displaced task to use the "new" resource without affecting or severely degrading the performance of the task(s) that were originally assigned to it.

Alternately, one might consider shared resources as the second type of redundancy – a variant on the virtual node clone scenario, in which redundancy is not expressed in terms of duplicate nodes, but rather as secondary resources that may or may not be suitable replacements to a displaced task. As an example of the second option, let us assume that a collective of ground-based, extra-planetary mobile robots is exploring an unknown terrain. Within the network, one robot provides a high-resolution color imaging capability via a charge-coupled device (CCD); this provides data in real-time to enable

obstacle detection. Another robot is designed to provide thermal imaging, which requires a low-resolution CCD camera onboard that robot. Should the robot with the high-resolution CCD fail, we can re-allocate the second, low-resolution CCD device on the thermal imaging robot to replace the first robot's camera. If the low-resolution CCD is acceptable, we say that it is a *fit* resource. It is possible that multiple fit resources may exist on a network. Our discussion now leads us to existing frameworks for identifying capable resources on a MANET. These frameworks are called Service Discovery Protocols, or SDPs, and are discussed in Chapter IV.

CHAPTER IV

ATTACK AND FAILURE MITIGATION THROUGH REALLOCATION

This paragraph aids the reader in understanding basic principles about system resources, their consuming tasks, and ways to search for additional available resources in the event of a node or resource failure. By taking advantage of available resource redundancy on the network, reallocation of these tasks to new resources can take place in a manner that is most-efficiently suited to the network's design needs.

Discovering Resources

Mian et al. [9] suggest that most service discovery protocols operate on the application layer in a manner similar to wired networks, with the caveat that service discovery on MANETs must operate with location and proximity-awareness in mind. So, what does an SDP do, given a set of nodes, mobility awareness and a consumer of the services? And more importantly, how does any of this relate to the continuing discussion on reliability and security of task-to-resource-allocated ad hoc networks? This section, *Discovering Resources*, will introduce principles of service-oriented SDPs and routing-oriented SDPs and then show how their principles can be applied to offer a solution to the *task-based* model called DARTS, implemented in Chapter VII.

Organization and Registration

According to Mian et al. [9], we must start with looking at a *service description*, which is a form of abstracting the qualities and capabilities of a service. A *service* is defined as a piece of hardware, software, or an abstract component that is needed by a

consumer (which can be high-level, such as the end-user, or low-level, such as another process or hardware instance). Examples of services might be a networked printer, data storage array, or an authentication service. Based on the service description, nodes on the network (wired or wireless) can then identify which node is a suitable candidate to fulfill those requirements. The method in which this is performed is specific to the algorithms implemented by the particular SDP. *Registration* then refers to how nodes on the network store information about the services represented by the MANET. The completeness of registration data on a global level dictates the depth of which a search algorithm must be performed to find valid resources. In addition, how much information is stored on each node also dictates whether or not nodes will need to perform periodic updates to refresh knowledge about the services provided by other nodes. At this point, we begin to see a tradeoff; in the extreme cases, there are two possible outcomes. The first case is where no node knows anything about the services of surrounding nodes. Should a consumer poll the network for a service, the network must be flooded with requests to find a suitable service. The benefits to this scenario are that no updates are needed among nodes. If a consumer arrives only at extended intervals, the elimination of updates may increase efficiency and power utilization on the network. Alternately, the second scenario dictates constant updates among nodes in such a manner that all nodes are aware of all the services provided on the network. This presents scalability, memory utilization, and traffic problems, especially when considering mobility. Scalability can be a problem because updates must propagate continuously among many links. As the network increases in size, the number of update transmissions will increase accordingly. Memory utilization – a significant issue on MANETs which may have limited memory capacity, is

used up by lookup tables and similar structures to keep updated copies of node resource information tables. These tables must be updated often if nodes are continuously associating and disconnecting from the network, and if the topological configuration of the network is altered due to the mobility of its nodes. Traffic issues arise from the need to send update information from one node to the next. Depending on how much data is being transmitted, this may be a non-trivial amount of overhead on a network, particularly 802.15.4 networks that have a low average bandwidth.

Resource Discovery via Network Flooding

Following registration, we can then perform a search for services, which is initiated by the user or one or more arbitration nodes. Active searches correspond to registration methods that do not have network-wide omniscience of resources. In this case, a flood of requests is sent across the network until one or more nodes return a match for the service. Alternately, registration methods that utilize global lookup tables or structures that are updated periodically through advertisement messages may reduce the overall traffic requirement at the moment of a service request. However, as mentioned earlier, this method is prone to long-term traffic problems as advertisements and updates must constantly be performed to conform to the network and resource configurations available at a given moment [8, 9, 13-15, 80].

Gao et al. [8] group SDPs into multiple approaches according to single, dual, and multi-layer approaches. In a single-layer approach, all nodes are grouped into one “logical layer” where each node provides the same category of services. According to the authors, flood search protocols belong to this single-layer approach. In order to reduce

the network overhead from flood searches, modifications to the flooding algorithm exist, such as the inclusion of probabilistic data to prevent multiple discovery routes from being re-used. Also methods exist in the single-layer approach for advertisement-base services, such as those mentioned previously. One such example is called the Group-based Service Discovery protocol (GSD). This protocol caches advertisements between nodes. Thus, service requests need not traverse the entire network to find a service, as a node further down the discovery chain may have knowledge of which node contains the service. Of course, this method is still prone to possible outdating due to mobility concerns, and node entry/exit. Furthermore, it still requires the implementation of advertisement packets, which create network overhead. Two-layer approaches utilize the clustering of nodes in terms of physical proximity to each other to reduce the amount of network traffic required to find a service. Hashing techniques are utilized in order to determine which node in the cluster (referred to by the authors as a matrix) is responsible for maintaining service information. Multi-layer structuring is a hybridization of several single or dual-layer techniques, and is generally too costly in terms of network overhead for use with MANETs [19, 81, 82].

Searching for Resources via Advertisement/Gossip

When considering implementation of SDPs with a MANET, special care must be taken for SDPs utilizing advertisement or gossip protocols. The rate at which a node moves over time will likely determine the topological arrangement. Thus, SDPs targeted at MANETs must incorporate feedback on mobility into both the registration process, as well as the discovery process. Note that an efficient flooding algorithm does not face these challenges as the state of the network during the flood search will be a *snapshot*,

assuming that the flood does not propagate too slowly. Thus, non-flooding methods must then update more frequently, using expiration data for the validity of the information in the registration tables. In addition to temporal modifications of the advertisement protocol, it is suggested that spatial modifications be made such that the *advertisement radius* be constrained to fewer devices. Again, this will have a long-term effect of requiring significantly more communications to achieve an updated registration [8, 13, 14, 20].

In addition to the service discovery protocols mentioned so far, there exist discovery protocols that are designed exclusively for routing purposes. Similar in nature to the SDPs for finding services, routing-based discovery protocols ensure that a network has routable links from point to point such that no node is isolated. Why should we look at routing-based discovery protocols when well-defined methods exist for SDPs? Routing on MANETs is a well-studied field that has yielded many accepted and standardized methods. AODV and DSR, for instance, form cornerstones of on-demand routing in which routes are not established until required by the network. Other strategies feature continuous updates, such as the advertising schemes seen for SDPs. There exist three main categories for routing protocols [22, 83]: Unicast, Multicast, and Broadcast. For simplicity and to stay relevant to future discussion, we will briefly cover the Unicast category here for perspective.

Unicast Routing

A unicast routing protocol is designed to allow for transmission of data from source to destination on a one-to-one basis. A node routes traffic from its origin along

various other connected nodes, referred to as hops. As a data packet proceeds from source to destination, information contained in the packet header is used to determine to which neighboring node the packet should proceed. Protocols such as TCP/IP utilize the unicast routing method. Unicasting is applicable to the vast majority of networks, which are based on relationships between producers and consumers of data.

Unicast routing algorithms can be subdivided into two categories governing their operation. The first is that of proactive routing. Proactive routing requires topological updates to occur periodically to refresh information contained in routing tables. This is similar to internet-based routing that occurs on traditional, hierarchical, general-purpose networks, especially those based on 802.3 and 802.11 operating in infrastructure mode. Proactive routing is similar in its update requirement as the advertising SDP registration mechanism. Since routes are accessed more frequently and more diversely than resource-to-consumer mappings found in SDPs, maintaining freshness of the routing tables takes on a new importance, and thus is likely to make proactive routing largely unsuitable for MANET applications. However, there exist some proactive methods which are designed to reduce the amount of traffic seen over the ad hoc network. One of these, the Optimized Link State Routing Protocol (OLSR) [26] uses extremely local flooding broadcasts of one hop in length. Nodes with special routing functions, called Multipoint Distribution Relays, are responsible for providing routing information to associated nodes in such a way that not all nodes need immediate knowledge of the large-scale network topology [22, 65]. Another approach, called the Topology Broadcast Based on Reverse-Path Forwarding Routing Protocol reduces the size of node and route update transmissions, and their frequency by only transmitting changes instead of complete, full sets of the

routing tables. In cases of significant mobility, the TBRPF method can switch to a more intensive flooding method to assure that all nodes are properly represented [19, 20, 22, 65].

Reactive Protocols

In contrast to proactive routing, *reactive* routing protocols are on-demand protocols that flood the network whenever a transmission is needed. This is useful in two cases: (1) for highly mobile networks, where flooding will, over the long run, consume less bandwidth than constant updates every time proximity relationships change, and (2) for networks in which the average rate of communication is relatively low. MANETs utilizing the 802.15.4-2006 protocol largely favor this routing protocol [22]. In the process of determining the route, two events take place. The first, route discovery, occurs as a result of needing to find the route. The second, route maintenance, occurs once a connection is established, typically over TCP, to prevent mobility from causing service discontinuities.

The most common implementation of the reactive routing protocols is the Ad Hoc On Demand Distance Vector Routing protocol, or AODV [84], developed by Nokia, the University of California Santa Barbara, and the University of Cincinnati. This protocol attempts to reduce overhead traffic and variable overflows in state storage that existed with prior proactive systems, such as DSDV. AODV sends a Route Request (RREQ) packet through the network by means of a packet flood. Upon receiving RREQ packets, temporary reverse routes are created from receiving nodes to the originating node. By definition, these routes are valid routes. Routes that resolve to the destination then reply

through these channels with a Route Reply (RREP) packet, sent in the reverse direction (i.e., back toward the originating node). RREPs also establish routes in the reverse direction. RREP routes that coincide with RREQ routes therefore form the basis for valid routes. Of these routes, the shortest (i.e., the route with the least number of hops or intermediate nodes between source and destination) is selected. Once the route is established, periodic maintenance messages are sent along the route. If at any point acknowledgements to the maintenance messages are not received, the route is invalidated. Should route recovery be unsuccessful in re-establishing the route, it is removed from the routing list. Thus, AODV fulfills routing discovery and maintenance objectives.

Multicast and Broadcast networks are high-traffic solutions to routing problems, and are optimized for systems in which one source or producer has multiple consumers. Some proposed methods include modification of AODV to broadcast route maintenance measures rather than single neighbor-to-neighbor updates, which would reduce the need for overhead associated with individual connections. Because of bandwidth concerns, this dissertation will not focus on multicast and broadcast routing methods as possible solutions to the task-to-resource allocation problem.

Now let us combine what we have discussed by analyzing similarities and differences between Service Discovery Protocols and Routing Discovery protocols. Both provide a framework for uniting one or more producers with one or more consumers on a one-to-one or many-to-one basis. In the case of an SDP, this is a service that is provided by a node on the mobile ad hoc network, and requested by an end user or another node on

the network. For routing discovery, a relationship is defined by network-layer routing that must be established between the nodes such that an originating node may establish communications with a destination node. The consumer, in this case, is the node requesting the route. Both types of protocols make clear distinctions between flood-based discovery and a maintenance or local broadcast strategy. However, SDPs and Routing Discovery Protocols (RDPs) have a distinct difference, which lies in the layer at which they operate. For the case of the RDP, routing is performed at the network layer. For a large part, this is a process that is completely transparent to the application layer and to the system in general. In contrast, SDPs focus almost exclusively on finding services at the application layer. SDPs assume the existence of connectivity and routing. Thus, an SDP may depend on an RDP, but not vice versa.

Tasks and Resources

Earlier, this dissertation mentioned some basic questions regarding the usage of SDPs and routing discovery protocols as possible solutions to the proposed resource allocation challenge for MANETs. Clearly, there exist multiple protocols that are mobility-aware, or can be adapted to mobility scenarios. As seen by SDPs, there already exist frameworks for finding services for nodes that are proceeding through the discovery process. What then, is the proposed research attempting to accomplish? The key difference between the method in Chapter IV and the function provided by SDPs is that in the proposed method, the consumers are tasks, which may span multiple nodes. An SDP matches an application-level service with an application-level consumer, such as a

notebook computer with a printer. To understand the difference, we must investigate the concept of tasks and how they fit into the picture.

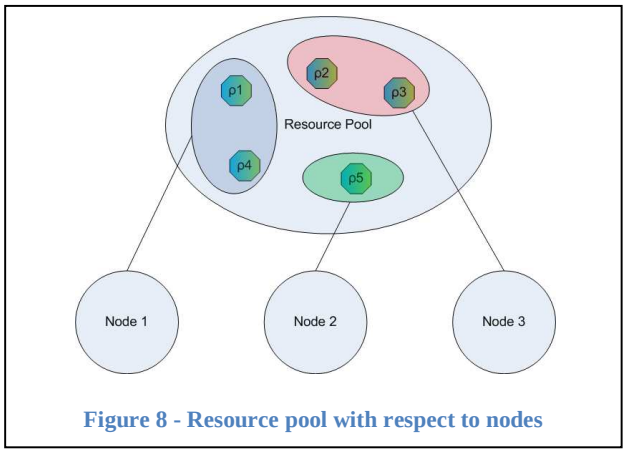

Figure 8 - Resource pool with respect to nodes

The term *resource pool*, π will now be introduced to define a set or subset of resources held or serviced by nodes on a MANET. A *global resource pool*, π_G, is the space containing all resource pools such that $\pi_G = \{\pi_i | \pi_1, \pi_2 \ldots \pi_N\} \forall\, i \in N$. There are two perspectives by which a resource pool can be identified. The first is the resource pools as nodes with respect to resources, and the second is resources with respect to nodes. This dissertation will use the second method primarily, as it allows finer granularity of the resources available to each node. Figures 7 and 8 illustrate the difference. For the most part, a pool π_i will be seen as a resource pool corresponding to one node *i*, while a pool $\pi_{i \cap j}$ represents a pool that is the intersection of resources provided by the two nodes *i* and *j*.

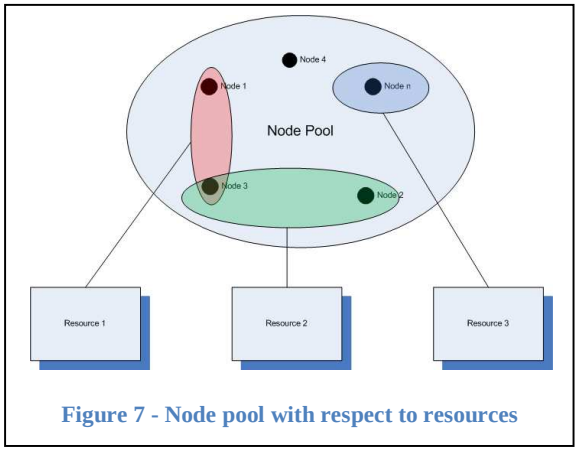

Figure 7 - Node pool with respect to resources

A resource pool allows us to efficiently understand the availability of resources at any given time *t* during the operation of a MANET. The proper visualization of such pools is essential to understanding how resource availability evolves during task

execution in a MANET. Resource pools may also be empty, in which case it will be referred to as a *null pool*, or $\pi_{\emptyset}$. This case represents an occurrence in which a node or nodes has no available resources at the given time. It may be the result of initialization, an error condition, or a termination state, or may simply represent a normal transition state. This dissertation will not use null pools as constant objects, since they represent inefficient or useless nodes that do not change over the course of the MANET's task execution environment, and are therefore not relevant. Null pools represent a node or a set of nodes in a non-interacting state with respect to resource allocation.

A resource pool π is comprised of nodes containing one or more resource elements ρ_i. Each resource is therefore indirectly a member of one or more resource pools. Because this dissertation will focus on node-level interactions and resource allocation, each resource will be expressed as a tuple $\rho_{(i,j)}$ where *i* indicates the resource number and *j* indicates the node to which the resource corresponds. In the case where a resource is present across multiple nodes (such as a shared resource), this will be identified with a brace in the second component of the tuple, containing the node numbers.

A resource is empirically defined as a hardware or software component that may be atomically applied to fill a basic need. Resources might include an imaging sensor, a thermocouple, a motor (though it and other resources are comprised of different elements, these elements are not within the scope of the application and therefore the resource is still considered atomic), a digital signal processor, a web server application, or a relational database. Each resource is used and implemented to solve an integral part of a problem.

Because this dissertation will focus on two problems, one of which is an allocation problem, we must consider the possibility of a resource that is desired or required but unavailable to any node. Such a resource will be referred to as a *null resource*, denoted by $\rho_{(i,\emptyset)}$ since it corresponds to the set of nodes containing the empty set. The null resource will be used during the theoretical and algorithmic solutions to the resource allocation problem. A *unity pool* is defined as a resource pool in which only one resource exists. By definition of a resource, this also implies that a unity pool also contains one node, as a resource must be represented by at least one node.

Tasks and resource mapping

A task is a process or consumer that requires one or more resources over time to accomplish an objective. An objective is an overall purpose that a system accomplishes. A task is a subset of an objective. As an example, an objective for the Spirit and Opportunity Mars Exploration Rovers might be to identify the presence of water on Mars. In order to accomplish this objective, tasks must be performed that lead to the desired conclusion or termination of the objective. For instance, soil samples must be analyzed with a Mössbauer spectrometer onboard the rovers in order to determine their composition. Prior to soil and rock analysis, the robots might utilize their Rock Abrasion Tool (RAT) to loosen samples of minerals from rocks and other formations. Each one of these operations is a task, *T*, such that $T_i \subset O \; \forall \; i \in N_T$, where *O* is the objective, and N_T is the number of tasks present on the system. A task is a comprised of one or more *subtasks*, s_i. The differentiation between a task and a subtask is important: a subtask is assigned to only one node, and therefore has access to a resource pool that is confined by that node only. Using the Mars rover example, the mineralogical examination represents

the main task. The subtasks are allocated to individual nodes; one robot might be responsible for using its rock abrasion tool and perhaps a manipulator, while another performs the analysis with the Mössbauer spectrometer and the alpha particle X-ray spectrometer [64]. (In practice, the current MER nodes operate independently. Their joint cooperation is listed solely as an example.)

A subtask is comprised of *atoms*. An atom is defined as the most fine-grained element of a subtask that can be mapped to one and only one resource. A subtask may be composed of several atoms, indicating that multiple resources on one node are responsible for executing the subtask. Alternately, the subtask may only be comprised of one atom. An example of an atom is the individual execution of the Mössbauer spectrometer's evaluation. Its operation is independent of the alpha-particle X-ray spectrometer. Given this terminology, we may now take a look at allocation and reallocation processes and proposed solutions.

(Re)allocation of resources

The allocation of resources represents a complete mapping of resources to tasks such that all the individual subtask atoms are fully populated by node resources. The problem presented by this work is two-fold. First, an *initial allocation* must occur, which provides the first mapping between tasks and resource pools in an efficient manner. This is an optimization problem, where an objective, comprised of tasks and subtasks yields a set of solutions depending on the level of redundancy of node clones, virtual node clones, or shared and share-able resources available at the beginning of task execution. The optimal selection of this solution depends on four main factors.

1.) **Spatial organization:** this parameter is used for optimizing task allocation such that nodes involved in communication between subtasks are physically proximate to each other. Two nodes that must communicate over multiple hops would cause overhead that might be avoided by having them be nearest neighbors.

2.) **Employing redundancy:** A well-designed MANET should be created with a certain risk assessment in mind. If it is estimated *a-priori* that the network will experience frequent node failures, attacks, or both, it may be useful to solve the optimization problem with a set amount of redundancy factored in. This can be achieved by increasing the number of resources and/or nodes, and then adjusting the spatial organization solution to accommodate the extra resources without creating unnecessary routing and communications overhead.

3.) **Traffic frequency analysis:** When considering an overall solution, let us briefly invoke the differences between flood-based SDP searches and node update/advertisement methods. The solution to the problem should reflect the amount of expected traffic on the network. Flooding might be more useful for low-traffic situations, while periodic updates are more useful for high-traffic networks, in which the updates will reflect a minor percentage of the actual communications throughput.

4.) **Mobility requirements:** Again, invoking the SDP discussion, understanding the degree of mobility between the nodes, and possible randomness of the motion is necessary to choose a protocol that is both resilient and efficient. The initial allocation may be performed *a-priori* and in a static manner, or dynamically according to a selected protocol or set of protocols.

Triggering the reallocation process

Once the initial allocation has completed, we can look at the main objective of this research: how to reallocate tasks to resources in the event of attack or node failures. Once a failure or attack takes place, a triggering mechanism must be in place to: (1) detect the absence or compromised state of a node, (2) begin the distributed reallocation process when appropriate or necessary, (3) suspend tasks that cannot execute due to incomplete or impossible mappings (resources that cannot be reliably re-allocated) and lastly, (4) re-trigger the re-allocation process upon either receiving newer replacement resources, or identifying that the failed or attacked nodes are once again fit to resume operation, in order to achieve an optimal allocation that is equal in performance to the initial dynamic or *a priori* allocation.

Given our previous discussion on intrusion detection technology (Chapter III), we can assume that triggering might be initiated by a distributed IDS engine present on multiple device nodes. At the point where anomalous behavior is detected, one or more nodes may trigger the removal of a compromised device. Once this occurs, depending on the reallocation algorithm in place, one or more nodes must then assume the process of regulating the discovery process. For some protocols, this is handled by a dedicated node. Of course, the downside to this is that the dedicated node may also fail, leaving the reallocation algorithm in trouble – even if more than one dedicated node exists. Alternately, a node may assume the responsibility on-demand, based on conditions present at the time. For instance, a node that is currently interacting with a compromised node may be the de-facto arbitrator of the reallocation process, as it is the first to need resources that are no longer available. There may also exist a further distinction between

the nodes responsible for identifying the compromised node and the nodes that perform the reallocation arbitration. Basu et al. base their system's triggering on a disconnect detection method; no IDS is present. The first node needing the resource or route becomes the coordinating node [80].

Reallocation Performance Metrics

Let us briefly consider some metrics that may be useful in determining the efficiency of a reallocation method. Some of these have been mentioned previously. The time to allocation is the first metric. This must be viewed in the context of other allocation methods as a comparison metric. The circumstances of the network's simulation should be identical. This metric is evaluated as a time in seconds to reallocation from the start of the triggering process until the replacement resource is engaged in execution. The second metric used here is network utilization, i.e., how many *messages* are required to instantiate the replacement resource? This dissertation chooses to use a message instead of packets in order to remove network-layer protocols from consideration. For instance, a protocol based on TCP/IP will inherently generate more packets for connection stability instead of one based on UDP. Thus, comparing them on packets would produce results that do not yield a representative comparison. A message is defined as a transaction from one node to another that contains information pertaining to discovery, resource requests, and/or discovered nodes.

Analyzing the number of messages leads us to the third metric: percent overhead. As mentioned earlier, the efficiency and performance penalties of a discovery search method must be viewed in terms of the overall communications activity on the network.

The same applies for a reallocation protocol. Reduction in the overhead ratio can be used to easily compare the efficiency of multiple protocol solutions.

There are two existing fields of task allocation and reallocation in existence as of the time of this writing. The first is that of Particle Swarm Optimization (PSO), proposed by Kennedy and Eberhart [85]. PSO represents a chaotic and randomized form of allocating consumers to producers by representing producers as objects on a two-dimensional grid. Spatial coordinates are representative of the organization on the network or system. The consumers are particles that move around the map in a clustered, yet chaotic manner. Each particle is given an extremely basic behavior, which is copied to all the various consumer particles. Because of randomization, all the particles behave slightly differently, but act together, similar to a flock of birds over a corn field [85, 86]. By utilizing these simplified behavioral movement models (often encoded by a single function governing its movement, expressed as one or two lines of high-level programming code), the system eventually settles on a solution.

PSO is not designed for use explicitly for MANETs. Since MANET nodes are generally operating as hard real-time systems, there is little room for a non-deterministic method like PSO. It is simply used as a reference for a randomized swarming allocation method. An extension to the PSO algorithm is proposed by Yin et al. that adds update models to add what the author calls exploitation – which can be seen as a further optimization of the solution once a solution is found [86]. The other model presented here is by P. Basu et al., called the Task Graph model [80]. It is a solution for service-oriented systems in which a consumer enters a network and requires a resource. The system can

detect disconnected communications using periodic update signals. Its algorithm requires network traversals and transmission of a task graph from node to node. The Task Graph model also identifies suitable replacement resources in case of node failure. It is capable of transmitting search requests through the entire network if necessary, and utilizes a shared information structure that is sent to neighboring nodes in order to reconfigure the network topology.

The shared structure in the Task Graph model is essentially a tree of nodes (i.e., system nodes) with dataflow edges that represent services between system nodes. The process of allocation and task graph generation is called embedding. Once a valid network representation is formed, identifying services and flow between nodes that offer them, a breadth-first-search algorithm is used to optimize the shape of the tree to reduce its complexity. The search is facilitated by local coordinator nodes. Figure 9 shows the Basu et al.'s representation of the embedding mechanism.

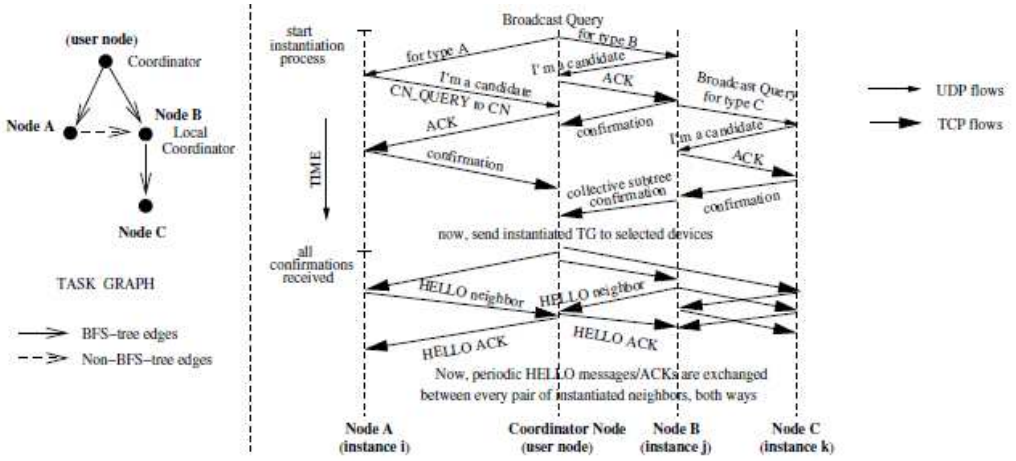

Figure 9 - P. Basu et al.'s Task Graph discovery method [80]

The Task Graph mechanism detects when a node disconnects if a node is unable to access a resource specified in the system model. At this point, the task graph must be reinstantiated, and the updated model must be retransmitted to neighboring nodes so that the system model remains complete. Figures 10 and 11 show the authors' visual representations of how this is executed.

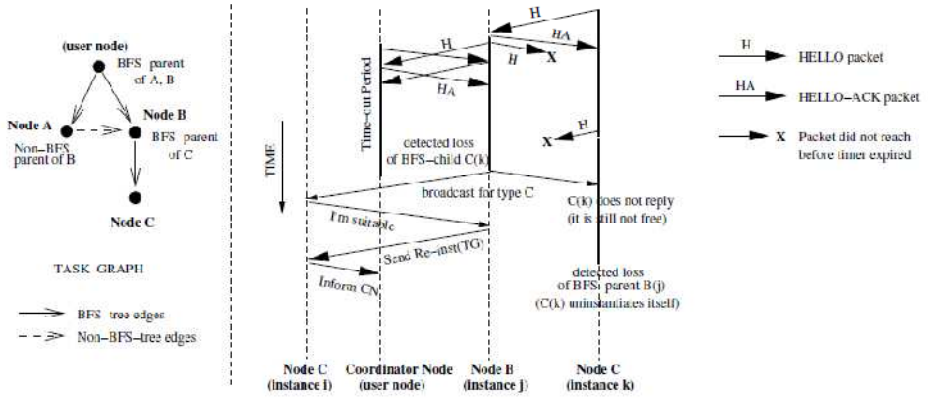

Figure 10 - P. Basu et al.'s Task Graph model service interruption [80]

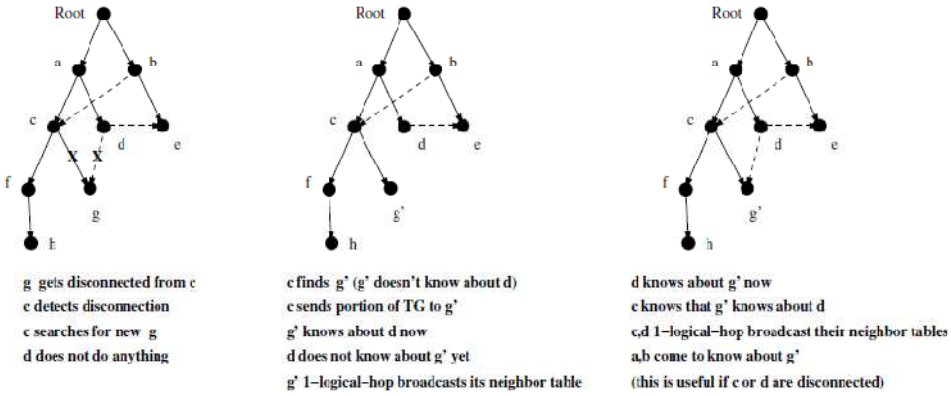

Figure 11 - P. Basu et al.'s Task Graph method node recovery [80]

Now that we have explored topics of security, fault tolerance, and discovery, let us turn our attention to ways of implementing discovery approaches into creating a fault-

tolerant networking atmosphere using techniques of resource reallocation, triggering, and principles of MANETs. The work in this chapter forms the basis of the fault-tolerant methodology [54] integrated into the developed system found in Chapter VIII. This scheme is called a Dynamic Apt Resource Transference System (DARTS) and is based on the concept of *resource fitness*. Let us analyze briefly what resource fitness is and how it fits into the scheme of DARTS.

DARTS: Fitness

With reference to the concept of a resource, described in Chapter III, a *fit* resource is defined as one that conforms to or meets the requirements of a task (or as a component of a compound resource) within a certain tolerance. For the sake of discussion, as well as for an analysis of the implemented methodology, the tolerance is represented by a maximum value in a set of integers from zero to n where at most $n = \max(\#nodes)$, or simply the value 9 on a scale from 0 to 9, so as to present a logical arrangement of fitness. These tolerance values are referred to as *fitness scores*, with a score of zero indicating that a node containing a relevant resource is completely unqualified to be a suitable replacement. An ideal fitness score is n. It is possible (and necessary to preserve equality, in some instances) for two or more nodes with relevant resources to share the same fitness score for their respective resources. This is especially the case on homogeneous networks in which nodes are identical in features and systems. Scores can vary not only based on the properties of the resources in question, but also based on proximity. For example, if a resource is sought and proximity is paramount, two identically-equipped nodes may have different fitness scores if one node is farther from the requesting node.

Resource Caching

As mentioned in Chapter IV, there exist two main methods of search discovery. The first is flooding [8, 9](also called on-demand), in which a node requesting a resource initiates a search that propagates to the rest of the network through a series of forwarded messages from node to node until the end of the network is reached. The second method, gossip [8, 9], requires that member nodes perform update requests from time to time between nearest neighbors (single-hop), which in turn updates routing and resource tables. Over time, information is distributed between nearest node sets and eventually, most nodes either contain or have access to a table wherein resources are properly defined.

The method investigated in this dissertation demonstrates a hybridization between the two processes, ideally combining the strengths of the two methods while drawing on both to minimize their disadvantages. By understanding a network at its application scope, we can identify components that help us to differentiate resources specifically within their operational concept, as outlined by the fitness scores. This is done through a component called the *resource fitness cache*.

As nodes interact on a regular basis, requiring resources from each other at either regular or irregular intervals, information about the resources being shared can be used to an advantage. Therefore, during the normal course of network interaction, each node populates and/or updates a fitness cache, which is a structure containing the resources and fitness scores of nearest neighbors (in this case, nearest neighbors are interacting pairs – other resource interdependencies may exist, e.g., between two non-directly-connected

nodes, but for the sake of cache coherency and freshness, these resources are not included in the cache) with information about the neighbors' resources and fitness scores. This method may appear similar to a gossip method, and in principle, it shares many common components. However, gossip protocols inherently have an update overhead – meaning that traditional gossip protocols consume network bandwidth with additional update information, not just data germane to the operation of the distributed task [8]. By incorporating information readily available during networked transactions and interdependence, the requirement for network overhead for the fitness cache method is eliminated. To avoid issues with stale data (i.e., data in the cache that is no longer accurate due to changes in network topology and resource availability), entries in the cache are aged. Beyond a certain age, cache entries are purged. However, because interactions are continuously occurring, the entries can be refreshed provided that an interaction has occurred recently to update an older entry. This ensures that frequent operations (and likely important ones due to their high level of occurrence) always have the freshest, most up-to-date entries in the cache. The purging age is determined dynamically during runtime, and is dictated by factors such as network size, mean message frequency, and degree of interconnectivity; primarily though, communication rate is the overarching factor that should be considered in network design. The aging protocol needs to be assessed for each target application; however, a general set of guidelines can be used: For critical systems that cannot afford invalid cache entries, entries should be purged if inter-node communication frequency has dropped below the mean communication frequency; If 20 nodes communicate at 4 Hz, and if a communications cycle is missed between a node pair, the cache should immediately go

invalid for whichever nodes have gone out of communication. Of course, for systems with irregular communications intervals, instead of using a unity scalar for number of absent communications cycles, this could be increased linearly. Ultimately, response time is based on the candidate application, and implemented by the IDS mechanism. Even if the caches are flushed too frequently to be of use in mitigating flood propagation, the resulting reallocation search cannot drop below the efficiency of the baseline, un-cached search.

Finding a Fit Resource

Now that the concept of fit resource caching has been introduced, let us then examine how DARTS proceeds to overcome node failures and reallocate resources. Figure 12 shows a network consisting of 5 nodes. One of the nodes, labeled 5, is drawn with a dotted line. This indicates that the node has either failed or is in the process of failing, either due to environmental concerns or a malicious attack. Its resources are 1, 5, and 7, and are represented in a blue table next to it. Tables in *green* are resource fitness caches. Note that the malfunctioning node also has a green fitness cache, meaning that it too was caching the fitness of other nodes' resources during the course of normal network interactions until it became compromised. The triplets in the cache indicate (node number, resource ID, fitness score) for each node's resources.

Next to it, Node 4 requires a resource that Node 5 was providing. At this point, there is a discrepancy between available resources and consuming tasks, assigned to the nodes. The minimum impact of this scenario, according to terminology defined in Chapter IV, is that an atom (an individual operation assigned to one resource on one node) can no longer execute, which therefore threatens the execution of a subtask (assigned to one node). At the task level, (assigned to more than one node) execution may not be immediately threatened due to sequential

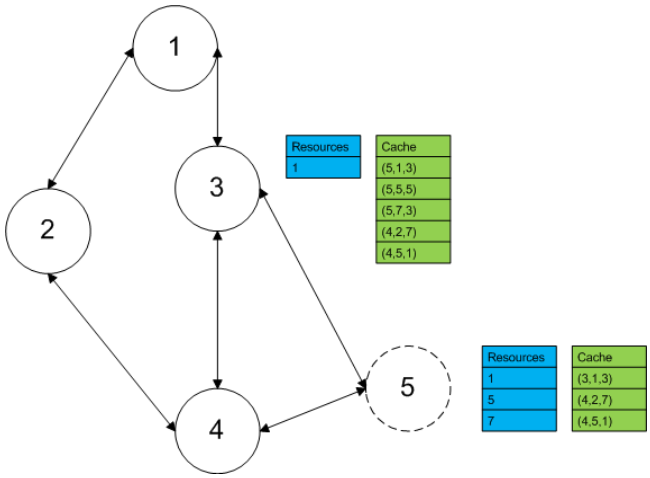

Figure 12 - A sample network with resources (blue) and fitness caches (green)

ordering of subtasks. In fact, a particular subtask may be called only as a contingency method, with a statistically improbable chance of being executed. However, for the sake of reliability concerns, it is assumed that any failure of an atomic operation and corresponding subtask is a threat to the operation of a general task, and therefore a threat to the overall objective.

Depending on the organization and population of the network, there is a reasonable probability that there exists resource redundancy somewhere on the network. For instance, a collective of autonomous aircraft may possess multiple ground-scanning radar modules, all of which may or may not be in use at the time of the resource disparity. The probability of a network having a possible replacement resource, therefore, is contingent on the size of the network and whether or not nodes are homogeneous (highest probability of a replacement resource), heterogeneous (moderate probability) or strictly heterogeneous (meaning that there is *no* resource redundancy – minimal probability of finding a replacement resource.) Note that in the strictly heterogeneous case, it is still possible to find a *fit* replacement resource, though it may not be an *identical* replacement resource.

To exploit the inherent redundancy in a MANET, Node 4, which was consuming a resource from the now-defunct Node 5, issues a flooding search request to determine whether or not fit replacement resources exist. The flood is begun with a message, called a Transference Inquiry Packet (TIP). Each TIP contains four information fields known as TIPlets: (1) the requested resource identifier (resources are categorized by integer types, which are known prior to run time), (2) the minimum required fitness score, (3) the

originating node identifier, and (4) the failed node identifier. Each one of these identifiers is used to allow receiving nodes to understand for what resources they should poll their cache, where a cache hit should be directed (the source of the flood's origin – the arbitrator), and, likely as important, what node is NOT available – the node containing resources that the originating node is attempting to reallocate. The failed node identifier issues a particular request directly to the resource fitness cache (called a cache manager) to purge the failed node from its cache. This way, cache coherence, in terms of removing invalid nodes, is assured throughout the network. Because requests are received prior to searching caches and forwarding requests, caches are purged without the possibility of a false positive. The node that initiates the TIP flood becomes the *arbitrator* node, and is responsible for managing the resource reallocation itself, once necessary information is returned from the rest of the network regarding available fit resources. An additional, optional fifth TIPlet includes a unique sequence number that corresponds to a particular occurrence of a resource search flood; all TIPs during the course of that search process will possess the same sequence number, which is then incremented once the next flood is initiated. This allows for proper distinction of different search TIPs, which aids in the mitigation of forwarded traffic.

When a node receives a TIP, it first checks its resource fitness cache to see whether there are up-to-date entries that may correspond to a neighbor with containing a fit resource, based on the criteria contained in the TIPlets. There are two outcomes to this search. A positive cache hit resolution indicates that the receiving node has identified a replacement resource, negating the need for further TIP propagation at that node to further nodes to which it is connected. Upon resolution of a cache hit, the node then

returns a notification of a found resource, and sends it by the network's routing protocols back to the originating arbitrator node, as identified in the originating node TIPlet. In addition, each node receiving a TIP stores the arbitrator's node number in memory, should these nodes eventually require the use of the failed resource; instead of issuing a new reallocation flood for each node that required a failed resource, the original search's arbitrator can be polled for the solution it found with an on-demand basis. The sequence number is important in this regard, as during the arbitrator polling process, the search's sequence number is used to inform the arbitrator which result to return, in case the arbitrator acted as the search initiator for multiple resource reallocations.

Alternately, should the node's fitness cache return a miss (i.e., indicating that either no matching resources are found, or that a matching resource exists, but of insufficient fitness required by the arbitrator node, and therefore the original task), then the original TIP is re-broadcast to all of the receiver nodes' neighbors, with the exception of the node from which the TIP was received prior to the cache search. Figure 13 demonstrates visually how cache hits and misses resolve. Nodes in green indicate positive cache hit, while nodes in red are nodes that have received TIPs, checked their caches, and found no suitable replacement resources among their neighbors as indicated in the cache. Blue nodes are nodes that were not reached during the reallocation flood due to caching efficiency gains. These nodes then forward the TIPs onward. In

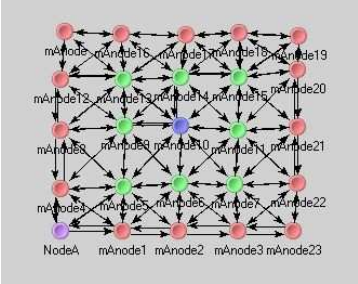

Figure 13 - A completed cached traversal

addition, in order to prevent re-broadcasting of TIPs to and from nodes that have already performed a cache search, nodes with cache misses become *insensitive*, meaning that further TIPs for that particular resource request will be subsequently ignored.

Once the arbitrator has received one or more returned resources within a timeout value, it begins assessment of the ideal resource based on replies. Although it may seem tempting to simply choose a found resource that has the highest fitness score, other factors must be considered before arriving at a final decision. Primarily, the arbitrator must decide whether or not a replacement resource is a sound choice, when considering network complexity. Therefore, the number of hops from the node containing a potential replacement resource back to the arbitrator is used to weigh the fitness scores. As an example, a node containing a replacement resource with score 6 (minimum 5) that is 2 hops away from the arbitrator (recalling that the arbitrator node is whichever node is first and foremost in need of the replacement resource) might be chosen as a more feasible replacement instead of another node with fitness score 7 that is 4 hops away. The system functions according to the following pseudocode example:

```
if ( detect resource failure)
        send TIP
        while (no replies)
                wait
        for all replies
                check for highest fitness score
                accept resource with highest score
```

The first justification for this weighting method might seem obvious – time and overhead mitigation. However, a second and more subtle point must be made that is in line with MANET resource constraints presented in Chapter II. Overall battery life will be significantly altered if more nodes are required to route information between two distant points. If instead, two resources are more locally positioned, then the traffic will require fewer hops, stressing fewer radios, and requiring less overall battery power.

In the event that the arbitrator receives no messages after the timeout value, the arbitrator must decide whether or not a subtask, task, or the entire objective must be suspended. The timeout value is determined by the maximum amount of time taken to send a packet over the longest possible path from the arbitrator to the most distant endpoint and back, plus a certain padding factor to account for high network traffic, cache lookup times, forwarding delays, and other node processing concerns. The level of granularity and overall impact of the unavailable and unreallocatable resource determines whether a subtask, task, or objective is suspended. For instance, if a collection of nodes is performing distributed sensing, and one particular resource is involved in optimization of the sensing efficiency, it is conceivable that the MANET could continue to achieve its objective despite a lack of replacement resources. Therefore, in this case, the optimization subtask (or task, if it involves multiple nodes) is suspended.

Task suspension must therefore be followed by a reactivation or reinstatement mechanism. This can occur automatically in one of two ways. The first case, in which a compromised node becomes once more available, may be triggered either by the re-

association of the previously-compromised node, or by a timed re-evaluation and re-issue of a search flood to see if a node is once more operational. Alternately, reactivation may occur if a new resource is added to the MANET, after which the announce and node discovery process allows the arbitrator to issue a reallocation and reactivation of the suspended task(s). Either case requires a reassessment of available resources on the network in order to achieve reactivation or reinstatement. In the case of a node becoming available again, the node will then issue the reallocation mechanism itself. Because the DARTS reallocation algorithm assesses the fitness of nodes, regardless of whether or not they are replacements or the original resource-containing node, the rehabilitated node will then assume control once more. Logically, this feature can be enabled or disabled based on security concerns of allowing previously potentially-compromised nodes to redirect resource utilization. By default, this capability is disabled. In this case, only when a new need for resource redundancy occurs, for the same resource, will the rehabilitated node be reinstated as the optimal resource solution. This permits DARTS to create deterministic reallocation schemes, assuring that given the same set of circumstances, the same reallocation procedure will be met, barring unusual conditions on the network.

Benefits and Limitations

Because DARTS is intrinsically uses a flood-based search algorithm, it is guaranteed to reach all network nodes if no cached entries are found. This means, from a reliability standpoint, that one or more, even all, of the resource fitness caches located on each node can fail to operate or be disabled without a loss in the capacity to reallocate resources. Of course, the main benefits of DARTS's caching protocols, namely message reduction and decreased reallocation time, are then no longer available. Therefore, the

availability and functionality of each resource fitness cache on each node on the MANET is critical to the prevention of unneeded TIP propagation.

This fact brings up the question of what happens when DARTS-based MANET implementations have nodes with incomplete or non-functional fitness caches. Because some caches will still be available, the number of messages is still likely reduced, but will still be higher than during optimal configurations. Ultimately, the position of a node with a working cache can determine much of the network's behavior. Consider the network shown in Figure 14. In this case, there exists a serial link between two network clusters. Supposing that node AC0 in the left-most cluster is attempting to find a resource located on node AC4, the cache of node AC3 is key to preventing message requests from forwarding to the entirety (or a large part) of the rightmost cluster. Because of this, we say that node AC3 controls a *critical path*. If, instead, the node were not part of a serial link, and instead an isolated fringe comprising part of a densely-clustered network, the benefits of DARTS's caching algorithm would be significantly reduced.

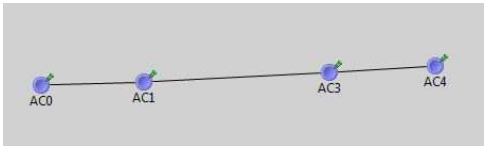

Figure 14 - A serially-linked network

CHAPTER V

NETWORK TOPOLOGY AND REALLOCATION EFFICIENCY

Clustering Density

This chapter analyzes how a particular aspect of network topology, known as clustering density, affects the speed and efficiency of DARTS. It begins by defining how clustering density is derived, and proposes how density can affect the performance of the system. Results are later discussed, and implications of other topological impacts on the efficiency of DARTS are analyzed.

This dissertation will analyze *clustering density* as a means of identifying the concentration and distribution of nodes. A segment of the network with high clustering density will represent many nodes in close proximity with significant interconnectivity among neighbors. A segment with low clustering density will instead be sparsely populated, with a higher likelihood of containing serial links. A serial link is a connection segment of network topology that has at most two nodes that it connects such that the loss of the this link would impair communications between the outer two nodes. Figure 15 shows an example of various clustering densities. It is important to consider clustering density as a criterion for selecting a reallocation protocol because the arrangement of the topology will affect how the discovery portion of the protocol operates. Densely interconnected nodes can, if flooded at a basic level, generate tremendous amounts of redundant traffic. On the other hand, serially-connected network segments can benefit from flooding, since it will guarantee network traversal regardless of other factors in the implemented algorithm.

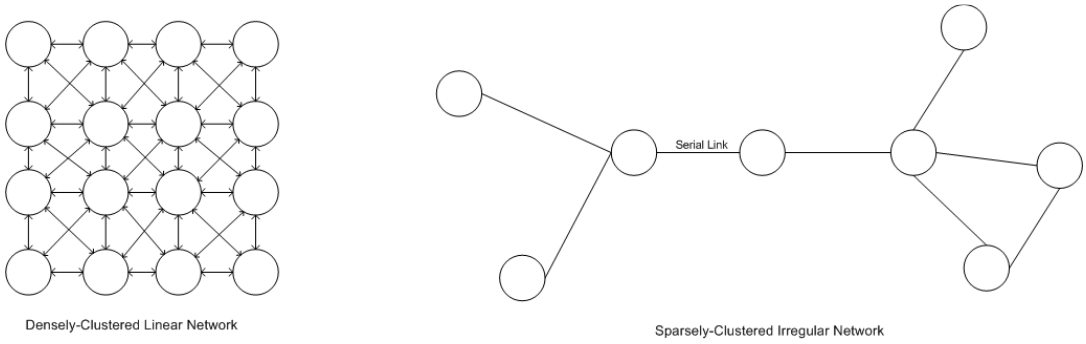

Figure 15 - Two examples of network clustering densities

Let us define clustering density as a visual and mathematical observation of the number of connections (unique radio links) as related to the number of nodes present on the MANET within a certain hop radius. Let us define a center node as N_c, where the center node occurs at the geometric center of the MANET. This geometric center is not strictly or rigorously-defined, but rather visually identified. It can be considered the point closest to which a network, if put on a two-dimensional plane, would balance; its center of gravity, if it were so quantifiable. Also, let us refer to $N_{(c+1,i)}$ as the i-th nearest neighbor node that corresponds to the center node, meaning that it is one hop (connection) away from N_c. We refer to $N_{(c+2,j)}$ as the j-th nearest neighbor of each $N_{(c+1,i)}$, such that each $N_{(c+2,j)}$ is one hop away from $N_{(c+1,i)}$ and thus two hops away from the center node. In addition to quantifying the nodes themselves, this dissertation defines the quantity L_c as the number of unique connections or *links* that are present between a center node and its neighbors. Similarly, $L_{(c+1,i)}$ and $L_{(c+2,j)}$ refer to the number of unique links present on nodes one and two hops away from the center node. It is important to understand that there cannot exist any duplicate links when all links L are

summed. This means that links counted on the center node L_c mutually exclude links counted from any $L_{(c+1,i)}$ that represent the same connection. Specifically, clustering density is expressed as a density quotient, Q_D, that is evaluated as:

$$Q_D = \frac{L_c + \sum_1^{i*} L_{(c+1,i)} + \sum_1^{j*} L_{(c+2,j)}}{N_c + \sum_1^{i} N_{(c+1,i)} + \sum_1^{j} N_{(c+2,j)}}$$

Due to the mutual exclusivity requirement in counting links, $i^*$and j^* are denoted as such with an asterisk to imply unique, singly-counted links. This is only true for the number of links; the number of nodes will always be unique. The computation of the density quotient outlined here is called the *two-hop density analysis method*, or simply the *two-hop method*. An alternative method does not restrict the counting to two hops, and instead uses a fractional-exponential weighting mechanism (nearby nodes are weighted higher than outlying nodes) – meaning that the density is computed similarly to the two-hop method, but instead of stopping at two hops, the entire network is included in computations. For each successive hop, individual link/node quotients are weighted differently and added to the overall sum. While this method has some advantages, such as covering the network extensively, it is subject to both computational difficulty, and unequal representation for networks that have skewed center node points.

Prior work was done to understand the effectiveness of DARTS on various levels of clustering densities, using a unified topology approach [54]. By attempting to minimize the changes present in the physical network topology, the research focused on understanding the gains that DARTS provides over baseline network cases. It also utilized the two-hop method for determining clustering density. The paper showed that

MANETs with higher clustering density quotients benefitted the most from DARTS, since having a working cache mechanism can significantly reduce the spread of messages to the rest of the network. An overall reduction of over 40% was shown in some of the densest cases. In addition, it was demonstrated that there was an appreciable decrease in the amount of time needed to perform a reallocation task, up to 25% on densely-clustered networks. This study forms the basis to demonstrate how understanding the network density can yield optimal improvements in needed reallocation.

The networks represented in this study yield average network scenarios; a worst-case scenario would be outlined by a network with only serial links. A best-case scenario, on the other hand, would be represented by any network layout in which the replacement resource is located within one hop.

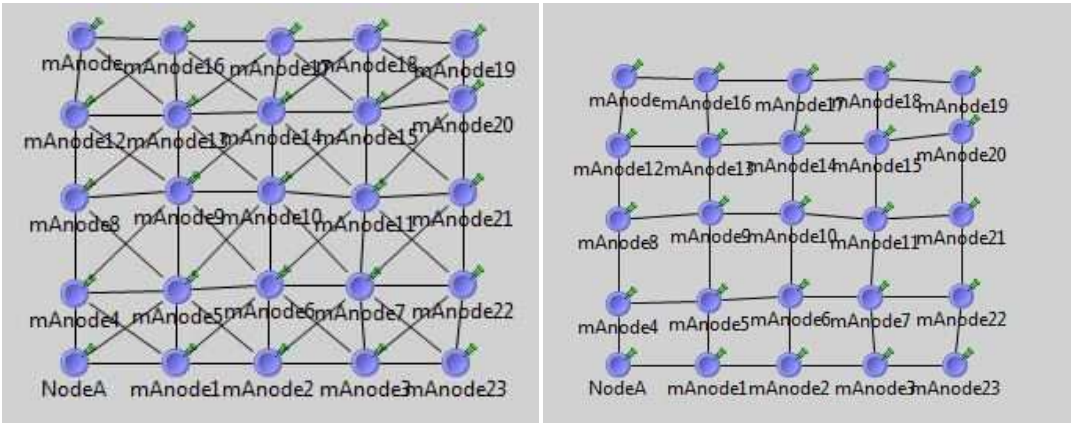

Figure 16 (a – left, b – right)- Networks with similar topologies but differing densities

Let us see how clustering density affects network throughput in a generic case. Figures 16 shows different networks with similar topologies but with disparate density quotients. The number of links present in the second network is reduced. From a logical standpoint, we could deduce that the second network presents a “less-dense” connectivity profile. Figure 16a shows a clustering density of 2.88, as computed by the approach mentioned earlier. Figure 16b presents a clustering density of 1.60 – a configuration in which nodes are at a “square” position with only links at normal angles (no diagonal cross-connections.) Figure 17 shows a network of density 1.28. In all of these cases, denser networks experience an improvement in reallocation times, as well as the number of messages required. Table 1 shows a list of clustering densities, the improvement in the number of messages, and the associated speedup. Note that there exist some unusual variations; one would expect a linear decrease in reallocation efficiency using DARTS as network density decreases, and more serial links are present. However, the discrepancy is likely in part due to undesired topological changes that will occur as the number of links is decreased; while a configuration may look like a square, it may not actually be a square shape when arranged in terms of logical links. This indicates that topology is an important factor in reallocation, capable of redefining performance in terms of expected values.

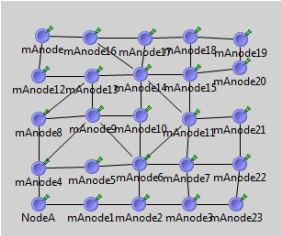

Figure 17 - A network with density of 1.28

Table 1- Raw cached and uncached results

Density	Baseline Messages	CachedMsg	BaseTime	CachedTime	%MessageImprovement	Speedup
2.88-inward	113	57	0.598	0.346	49.56	1.73
2.88-outward	118	103	0.685	0.356	12.71	1.92
2.64-inward	103	63	0.609	0.374	38.83	1.63
2.64-outward	106	93	0.430	0.226	12.26	1.90
2.56-inward	101	73	0.918	0.498	27.72	1.84
2.56-outward	103	90	0.894	0.597	12.62	1.50
2.52-inward	98	55	0.722	0.291	43.88	2.48
2.52-outward	100	88	0.621	0.376	12.00	1.65
2.24-inward	83	53	0.526	0.265	36.14	1.99
2.24-outward	86	75	0.611	0.343	12.79	1.78
1.92-inward	85	62	0.900	0.612	27.06	1.47
1.92-outward	87	80	0.783	0.687	8.05	1.14
1.60-inward	53	41	0.969	0.753	22.64	1.29
1.60-outward	55	51	1.068	0.794	7.27	1.35
1.44-inward	46	38	1.122	0.826	17.39	1.36
1.44-outward	48	46	1.285	0.839	4.17	1.53
1.28-inward	54	40	0.770	0.597	25.93	1.29
1.28-outward	55	51	0.982	0.601	7.27	1.63

Conditions marked “in” (shaded in light orange) refer to tests where a reallocation request originated at a fringe (near the outside of the network), while “out” refer to allocation requests originating from the geometric center and radiating outward. For purposes of discussion, and for data that makes more sense in terms of the application described in Chapter VI, we will analyze the “in” results. The “in” networks were generated by selecting the node at the geometric center because this would generate the

most traffic from a reallocation flood. Experimental results indicated that locating the start point for the “out” reallocation benefitted more from topological concerns than from the efficiency of DARTS as directly related to clustering densities. In order to have all network simulations be identical in their staring points, the center starting point was preserved. Figure 18 shows the improvement in the number of required messages for the “in” case, and Figure 19 shows the related speedup.

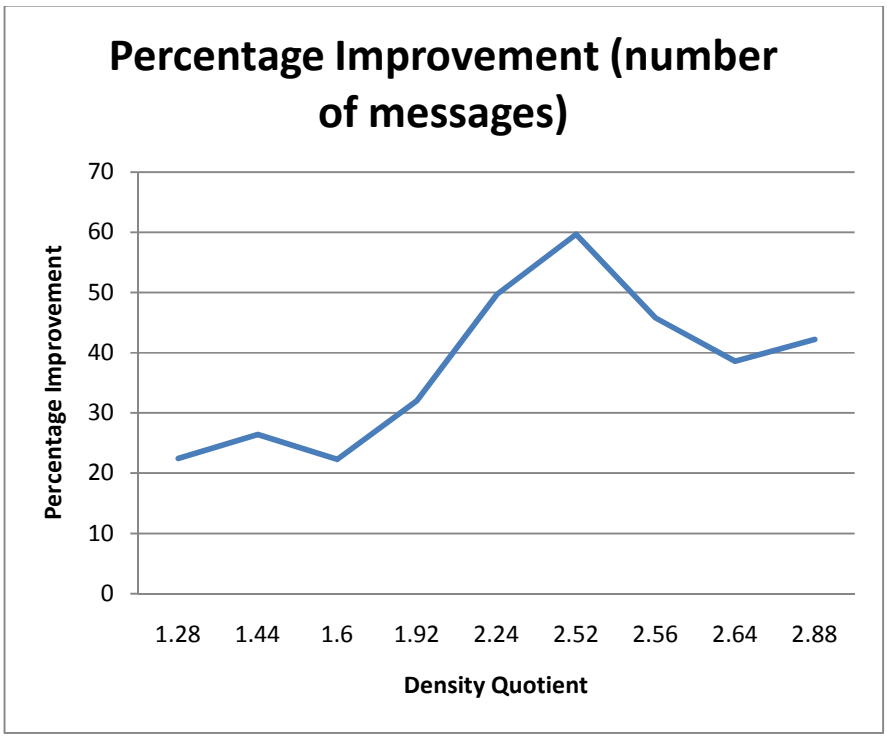

Figure 18 - Improvement in number of messages using DARTS over a variety of network densities

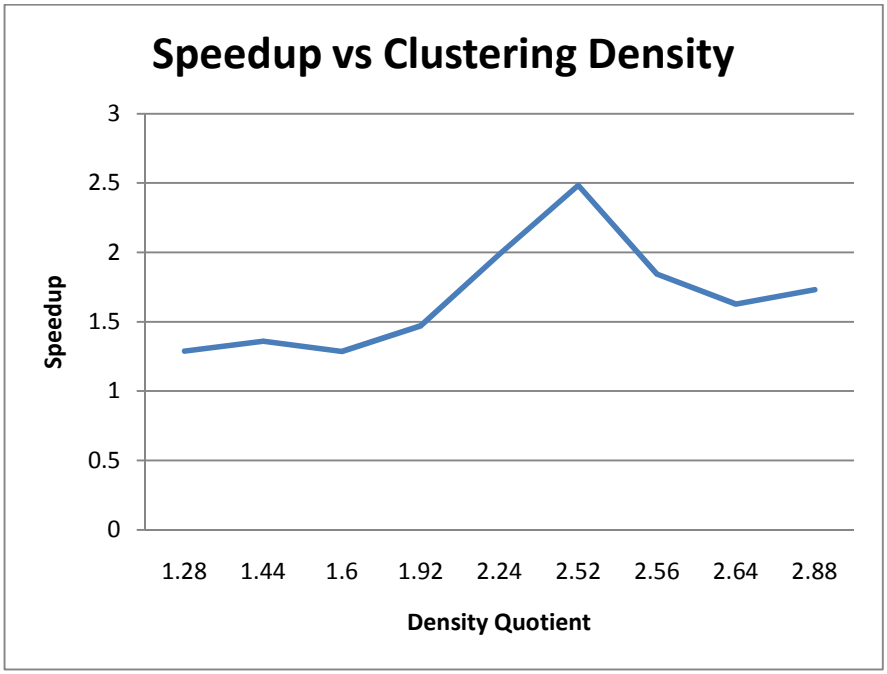

Figure 19 - Improvement in speedup using DARTS over a variety of network densities

While some of the results are unexpectedly non-linear, topological concerns seem to alter these results. If some of the unusual results are discarded, there appears a general linear trend from minimum to maximum density. The reasoning for the unusual peaks is simple: although serial links are generally conducive to worse performance, in some cases, networks will experience certain types of serial links that actually stem the propagation of TIPs to other portions of the networks. When examining the networks showing a higher response, this is the case.

Combining DARTS with a Triggering Method

As mentioned briefly in Chapter III, a holistic approach to security is preferable to any method involving only one aspect of computer security; if a system only decides to implement passive encryption methods, it will be vulnerable if or when sufficient techniques are developed to circumvent this static technology. Alternately, systems that

are only based on intrusion detection technologies are vulnerable because passive encryption plays an important part as a deterrent to intrusions. And, of course, a system based only on intrusion response cannot exist without at least the detection component. For this reason, this dissertation advocates a combined approach to intrusion prevention that integrates passive encryption (which for reasons of scope is assumed to be functional, even if imperfect) with an intrusion response methodology – a resource-aware intrusion detection technology that serves as a triggering mechanism for the resource reallocation method that we have outlined in this chapter. Such an integrated approach offers several benefits, such as being able to detect and correct for attacks and failures before they cause system downtime due to resource or node failure. In addition, detection can be used as a tool for optimizing system design, allowing network failures to be quantified and applied towards improving the network's connectivity and methods of operation.

Primarily, there exists overlap between the cryptographic/obfuscation methods and the intrusion response mechanism. Therefore, the failure of either system is in itself not conducive to imminent systems breach, unless the failures are connected (e.g., as the result of a sophisticated attack). Secondly, encryption and detection/response operate on distinctly different components of the host system. While encryption serves as a standalone component designed to work with the communications infrastructure, intrusion detection and response are application-aware and thus work on a different level independently of the passive prevention phase. From a fault-tolerance perspective, this is preferable both from the security perspective (i.e., neither systems have shared data, preventing an attack from simultaneously disabling both components) and from an

error/failure mitigation perspective (i.e., a fault or bug in one system, due to lack of shared data or global variables will not affect the stability of the other, barring a kernel panic.) Lastly, overlap is useful in case one system or another is unable to protect or identify a threat; the probability that an attack both disables an encryption method AND is capable of going undetected by an IDS is improbable (with the exception of internal malicious modification of runtimes or compiled source code).

Limitations of the Triggered DARTS Approach

Any good developer will freely acknowledge potential limitations of a system, and this author is no exception. The reallocation method, for instance, is dependent on the triggering method selected. In the case of this dissertation's intended implementation, triggering is based on two components. The first method examines both radio power level monitoring and packet retransmit requests. By monitoring dropped packets, and the subsequent modification of the radio's power level to attempt to compensate, one of two things is likely happening: (1) a physical barrier or range limitation is occurring, and one of the two nodes will soon drop communications, or (2) the network is being jammed. By monitoring subsequent increases versus packet drops, a basic profile and signature can be assembled, which can then trigger reallocation.

The second triggering method implemented is based on the author's prior work in lightweight distributed intrusion detection technologies. In this case, we can identify statistical events related to the system's application layer. In [28, 29], which describe the multi-stage IDS called HybrIDS, there exist two complementary methods for detecting anomalies at the application layer. Instead of relying on signature-based detection

schemes in which compute-heavy packet analysis is necessary, application-layer scope allows the IDS to reduce its dependency on data collection and, like the reallocation method specified in DARTS, is based on device-to-device interaction.

In the case of a triggering implementation using HybrIDS, there are some unique benefits as opposed to utilizing other IDS methodologies. First, as mentioned before, both DARTS and HybrIDS use node-to-node interactions to either monitor system activity (and analyze it for possible deviant behavior) or to update resource fitness caches to improve the efficient search for replacement resources if a node or specific resource is lost. This has far-reaching implications because it does not require duplication of data collection in the effort to secure and harden the host node's system. Furthermore, both systems are node-based, meaning that they are decentralized; the same data set that is relevant to HybrIDS is also directly relevant to DARTS. With this information, it seems logical to pair HybrIDS as the triggering method to initiate reallocation by DARTS. In fact, while the two are different systems and, other than sharing a common data stream, are isolated from each other, they can operate in one overall process or security protocol suite, managed with independent execution threads.

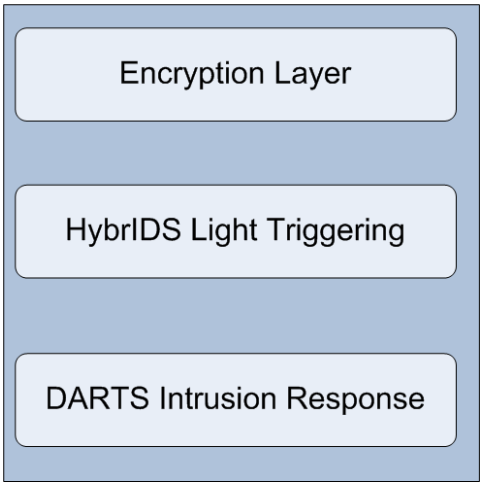

Figure 20 - A Layered Response Mechanism

A common dataset also allows for information exchange between the IDS and the reallocation system – more than a simple reallocation triggering request. Both HybrIDS and DARTS use similar storage methods for their per-

node databases, meaning that future functionality could be added that allows DARTS to "peek" at results from HybrIDS to monitor certain network health conditions more closely. Also, because both systems are non-resource-invasive, they can operate without overwhelming the host node. Because computational effort involved in collecting data can be combined and shared, the amount of unnecessary polling overhead is reduced by 50%, which would not be the case were heterogeneous, non-complementary technologies implemented.

Because of this integration strategy, there now exists a viable three-stage solution covering encryption/obfuscation, intrusion detection and intrusion response components (Figure 20). The encryption method is assumed to be functional (as per secure communications protocols that have been established in the system communications specifications). Also, the IDS and response layers (HybrIDS and DARTS) are designed to implement the detection and response components efficiently and interoperatively, yet with sufficient component-based autonomy. This requirement avoids issues if one system or another were compromised due to either security or system fault concerns. Now that we have explored this combination, let us see how it can prove a worthy part of a real-world implementation with a unique distributed sensing application, namely disaster area assessment with UAVs, which is also a contribution of this research.

CHAPTER VI

TORNADOGENESIS AND CLASSIFICATION

According to data gathered in the calendar year 2008, over 1690 tornadoes struck the continental United States [87]. These wind-based phenomena are capable of creating severe destruction in localized areas with which they make contact. Although hurricanes cause more widespread devastation, there exists no more intensely-destructive natural event than a strong tornado. Understanding tornadogenesis is a complex and multi-faceted field, requiring the computation of tremendous amounts of data. Despite all the work that has been done in over decades of research, comparatively little is known about how, why, or when such storms form. Understanding storm aftermath is a critical piece to unlocking the complex problem of tornadoes. For this reason, this dissertation implements the DARTS resource reallocation mechanism in a unique way, with the intention of aiding first-responders and meteorologists in aiding victims and understanding the storm's effects, hopefully yielding greater prediction accuracy and furthering the understanding of these elusive storms. This section will briefly outline tornadogenesis, storm characteristics, and storm classification that will aid the reader in understanding this research's application of DARTS.

Tornadogenesis

The fundamental principles behind tornadogenesis are well known, particularly for supercell thunderstorm types. While squall lines (resulting from the clash of cold and warm weather fronts) are capable of creating tornadic storms, these are generally rarer and less intense. Therefore, this discussion will focus on tornadogenesis within the

context of supercell thunderstorms. A supercell is defined as a storm, typically localized within a single draft zone, containing the rotation of winds within its cloud base [88]. These storms are typically accompanied by hail, torrential rains, and frequent lightning. There do exist low-precipitation (LP) supercell storms, which, by definition have little or no rain associated with them, but this discussion will focus mostly on high-precipitation (HP) supercell storms, which are the most common. Another factor is the atmospheric instability, which measures the vertical temperature gradient. Higher instability implies a stronger gradient, which creates a stronger convection potential. A high atmospheric instability combined with moisture form the basic elements needed for supercell storm formation. Regions in the United States that include large, flat plains, offer the perfect conditions for thermals to form updraft columns. In addition to heat, these regions are located in such a manner that moisture from the Gulf of Mexico can often interact with the thermal activity and form powerful storms that have the capacity to be tornadic (spawning the term Tornado Alley for states where this combination of moisture and thermal activity is optimal). Once atmospheric instability interacts with warm, moist air, a vertical convection potential is created to cause that warm, moist air to rise into the atmosphere. On a calm day, this occurs rapidly, and updraft currents can quickly bring the moist air up to high altitudes where it condenses and forms clouds. With stronger gradients, updraft currents can be exceedingly fast. In typical thunderstorms, such updrafts average between 1,200 to 2,500 feet per minute, roughly between 15-30 miles per hour [88] (the National Oceanic and Atmospheric Administration, or NOAA, reports its data using the English measurement system). Supercell thunderstorms, in contrast, have updrafts that can reach up to 15,000 feet per minute, or approximately 170 miles per

hour. These large and powerful air columns create a significant pressure differential at the base of the updraft column. In contrast to the updraft column, which pulls in warm moist air into the storm, a *downdraft* column represents the flow of condensed, cooler air that falls downwards, much like the gel-like medium in a "Lava Lamp." The downdraft column occurs at the "front" of a storm cell, along with any precipitation phenomena. Figure 21 shows a classic supercell thunderstorm with precipitation phenomena. Downdraft phenomena are attributed to devastating straight-line winds, which can flatten large swaths of land with downward-directed winds exceeding 70 miles per hour. These straight-line wind phenomena are typically called micro or macro bursts, depending on their size and speed [89]. Often, damage that is initially attributed to a tornado can in fact

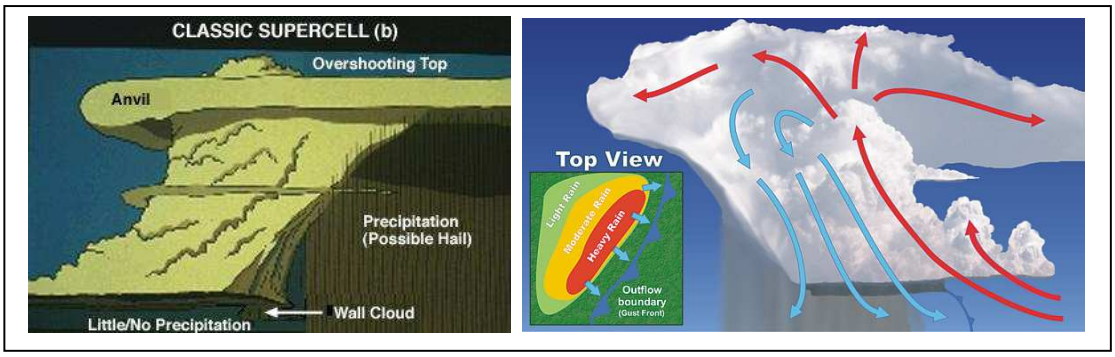

Figure 21 - A classic supercell storm (left) and the updraft (red) and downdraft (blue) columns (NOAA)

be the result of a downdraft phenomenon. Upon closer inspection, the direction of *fallen debris, and the wide damage pattern can, to the* experienced observer, confirm whether or not a tornado or downdraft phenomenon has occurred [88]. The technologies presented in this dissertation can be of further assistance in determining such differences.

Following the formation of a strong thunderstorm, and an established updraft/downdraft in the storm's core, further conditions are necessary for storms to spawn tornadoes. If the atmospheric temperature gradient is sufficient, and the main

updraft zone is strong enough, two factors may (with relatively low probability) contribute to tornadogenesis. The first is the formation of a mesocyclone – in which the updraft column begins to rotate. This is generally the result of wind shear – either from weather boundaries (the meeting of hot and cold fronts) or inflow phenomena that cause horizontally-rotating air movement. In supercell storms, these rotating columns are affected by the updraft zone at the core of the storm, and set in motion.

Figure 22 - A well-defined cloud at the base of a supercell storm

This causes rotation around the updraft zone, which can extend below the cloud base into a funnel cloud. Prior to tornadogenesis, moist air that is cooled by rain causes condensation to occur below the cloud base, forming a *wall cloud*. Figure 22 shows a storm base with a well-developed wall cloud.

The second phenomenon associated with tornadogenesis is less-well understood, and is called a rear-flank downdraft (RFD). Unlike forward-flank downdraft, which is considered the “normal” storm downdraft, RFD occurs behind the storm, and is a closed-

loop rejection of moist, warm air entering the storm's updraft zone. RFD is responsible for the formation of the wall cloud, as well as a "clear slot", which is a precipitation-free, and cloud-free area directly behind the wall cloud and related storm. It is postulated that RFD is crucial both in tornadogenesis, as well as in a tornado's eventual demise, as strong RFD tends to eventually wrap around its tornado, severing the flow of vitally-needed warm moist air that fuels the strong updraft zone.

As mentioned before, atmospheric instability, in addition to warm, moist air is essential for storm growth and eventual tornadogenesis. Wind shear, which can, with an updraft, create a mesocyclone within a storm, is also responsible for mitigating or limiting the growth of storms. If wind shear is present at high altitudes, caused by high-altitude air currents such as the Jet Stream, updraft zones in storm cells are cut off and shifted, tilting the precarious updraft/downdraft balance and effectively destroying the storm's potential energy and growth. Figure 23 shows thunderstorms being affected by strong upper-level wind shear – practically decapitating the storm's top. Given sufficient shear, storms cannot form updrafts strong enough to sustain tornadogenesis.

Figure 23 - Storm clouds under strong shearing conditions

Anatomy of a tornado

This section has briefly described the conditions and storm types necessary for tornadogenesis. Let us briefly discuss tornado anatomy. For the sake of clarification, a tornadic storm product is referred to as a *funnel cloud* if it has **not** made contact with the ground. A funnel cloud that begins to stir up debris from the ground is then called a tornado. Tornadoes are generally asymmetrical with respect to a horizontal axis, meaning that the part of the storm descending from the wall cloud is typically thicker than the end of the funnel cloud touching the ground. The "cloud-like" or opaque nature of the tornado itself is due to the condensation of water caused by the extreme low-pressure differential created in the tornado's updraft column, aided by the rotational movement of the mesocyclone that keeps the cloud formation cylindrical. In the case of weaker tornadoes, where the updraft pressure gradient is lower, condensation may not occur at all, yielding weak tornadoes that stir up dust, but not much else. Despite the absence of a visible

Figure 24 - A classic stovepipe tornado

funnel, these are still characterized as tornadoes due to their nature and formation from a thunderstorm cloud's mesocyclone. Figure 24 shows a stereotypical tornado.

Concerning shape, there exist three main types of tornadoes, though many shapes and sizes exist throughout a continuum.

1. The "classic" tornado shape, illustrated in countless textbooks and imprinted into the minds of many, is called a stovepipe tornado. These funnel clouds feature a relatively-smooth, straight shaft that is proportionally not much wider at the cloud base than at the ground. Such storms indicate a well-formed and stable updraft zone in the supercell storm that spawns them.
2. Alternately, a wedge tornado exhibits a funnel diameter at the storm cloud's base that is proportionately much wider than the diameter near or at the ground. As wedge tornadoes can exceed over two miles in diameter, their cloud-base diameter may be so large that it appears to be low-hanging clouds strafing the countryside – a potentially deadly mistake in cloud classification.
3. Lastly, rope tornadoes do not fit either category, and are almost sinusoidal, or S-shaped from top to bottom, indicating a weak or unstable updraft zone that is shifting. Rather than descending in a straight line to the ground, rope tornadoes have dissociating, weak updraft columns. In fact, some rope tornadoes may appear to be "missing" entire sections of the funnel cloud as it progresses from top to bottom.

Figure 24 shows images of the three classic tornado types (stovepipe, wedge, rope). In addition to shape types, tornadoes either occur as single-vortex storms, in which the updraft column of the thunderstorm supercell extends uninterrupted all the way to the ground, or multi-vortex storms, in which the updraft column is comprised of smaller, but intense pairs of rotation (some storms have been observed to have five vortices or more). The first multiple-vortex tornado was discovered (and photographed) in or around 1957. The reason for this occurrence is still not yet well understood as of the time of this writing, but is being actively researched. Wedge tornadoes often will be multiple-vortex tornadoes. Multiple-vortex tornadoes should not be confused with *sister* or *satellite tornadoes*, which are two or more tornadoes occurring in the same storm or storm system, but resulting from individual and distinct updraft zones.

Figure 25 - Stovepipe (top left), rope (top right) and wedge (bottom) tornadoes

Tornadoes vary in color based on the medium that they are disturbing, in addition to diffraction effects caused by different angles of lighting. Some storms may have poor lighting or may be wrapped in rain (as is the case with high-precipitation supercell storms) and thus are exceedingly dangerous as visual identification is nearly impossible. Unknowing motorists may be unaware that a rainstorm they are approaching in fact contains a deadly funnel cloud. Vehicular encounters remain one of the leading causes of tornado-based fatalities, as automobiles may be lifted by a strong storm and tossed

elsewhere by the rotating winds [88]. Furthermore, the glass windows in a car or truck do not offer adequate protection against tornado-borne projectiles and debris (sometimes, inaccurately, termed *missiles*). The NOAA in past years has suggested that occupants of a vehicle seek shelter by exiting their vehicle and assuming shelter in a low-lying ditch. However, based on new public safety statements, and, in this author's opinion, common sense, this recommendation has been retracted, and motorists are urged to drive away from the storm at right angles if possible. This change is motivated by the fact that low-lying ditches do not provide adequate protection from large debris. Debris are, by far, the leading cause of injuries and fatalities due to tornadoes. The most violent of storms are capable of hurling large pieces of wood, brick, or steel at exceedingly-fast speeds. The results of these high winds should not be underestimated, as their effects result in situations ranging from the tragic to the bizarre – residents of storm-damaged communities have noted blades of grass shot like arrows through telephone poles, and even glass windows.

In addition to characteristics of wind speed, size, shape, and color, tornadoes are also characterized by their duration. An average, typical spring-time tornado generally lasts for less than five minutes. There is a general correlation between the strength of a storm and its duration; stronger storms have larger associated thunderstorm supercells, with larger, well-defined updrafts. The formation of such storms requires immense amounts of energy (in the form of heat) to remain viable, and thus the energy in the atmosphere required to generate such a large storm is also capable of sustaining it for extended periods of time. Note that tornadoes rarely maintain perfectly-constant contact

with the ground. Instead, they tend to “hop” along, destroying entire sections of communities, but sparing houses just meters in between.

Classification – The Fujita Scale

The classification of tornadoes is done by quantifying the amount of damage observed on the ground, rather than through recorded wind speeds using a Doppler RADAR. This quantization has been historically measured through the use of the Fujita-Pearson Scale, developed by Tetsuya Fujita and Allen Pearson at the University of Chicago in 1971 [90]. The scale quantifies levels of destruction of man-made and naturally-occurring objects, such as houses, trees, and strong brick buildings. Finalized in 1973, this scale lists a storm’s destructive power on an integer scale from zero to five, demarcated with the prefix ‘F’. The lower score on the scale, F0, indicates a storm producing little or no damage. In contrast, an F5 storm indicates a violent storm that destroys and levels entire communities with catastrophically-strong and fast winds. Tornadoes generally have inconsistent damage patterns; many tornadoes will “skip” or “hop” along the ground, rather than maintain constant contact with the ground. Because of this, damage is variable; some structures may be completely demolished, while others remain unscathed. For this reason, the maximum-observed storm damage to any structure is used to categorize the tornado on the Fujita scale. Wind speeds are, in fact, correlated to the various F-scale degrees, but offer only an approximate range, and can be deceptive.

An F0 storm is typically carries with it winds of between 64-116 km/h. However, because the F-scale does not derive its classification from observed wind speed, it is possible for a tornado with 500 km/h winds (measured rotationally) to register as an F0

storm if it does not reach the ground and cause damage. High-speed funnel clouds may form, but be classified as F0 storms for their lack of contact or damage. F5 storms, on the other hand, must have made contact with the ground in order for destruction to take place. These storms can be more accurately correlated to observed wind speeds. Utilizing the original Fujita scale, F5 wind speeds are characterized as occurring between 419-512 km/h (again, measured rotationally.) F5 storms are typically characterized by catastrophic damage to structures and complete devastation. Large objects, including automobiles, and in some cases, train cars and locomotives, can be hurled distances exceeding 100 m. Such storms can have long life spans (the longest-recorded F5 storm in history is known as the "tri-state tornado"), which remained on the ground for an uncharacteristically-long 24-hour period. An average tornado typically has a lifespan of shorter than 5 minutes.

In addition to observed damage and wind speeds, the Fujita scale also incorporates storm statistics such as storm track length (the distance over which the tornado was making contact with the ground), and path width, though this data is not used to compute the actual intensity score. Some of the strongest tornadoes on record were merely 150 m across. On the other extreme, the author has personally been involved in a tornado that struck his grade-school transport van on a return trip from an outdoor activity. The width of the tornado was approximately 0.8 km, but its wind speeds (128 km/h) and overall lack of destruction with the exception of broken tree limbs and scattered small debris rendered the particular storm an F0. Note that because of a tornado's horizontally-asymmetrical shape, the width of the storm is generally larger at the base of the cloud than on the ground. This is especially true for "wedge" tornadoes, which have widths at the cloud base exceeding over 2 miles in diameter. In some rare

cases, especially when a tornado is dissipating, the cloud-base portion may be at times thinner than the ground-based portion. This does not, however, indicate whether or not a storm is weak or strong, only that it is potentially unstable and at risk of dissipation.

Table 2 in the next chapter lists the degrees of the Fujita scale, from F0 to F5. Note that the Fujita scale, originally developed from 1971-1973, has now been replaced by the Enhanced Fujita scale, which was first implemented in 2007. The change to the Enhanced Fujita, or EF scale, was brought about based on studies of storm damages over a period from 1950 to 2000. The work done on this dissertation incorporates research done on 40,881 storms from 1950 to 2001, classifying each storm based on observe damage, path length and width, and wind speed. Each classification is statistically analyzed using a Weibull distribution of those parameters to form an average statistical representation of "typical" storms in each F-scale range.

Because the EF scale has only been implemented since 2007, it provides relatively little available data for use in this dissertation. For this reason, and because it has been established for such a long time, all references to tornado intensity will use the original Fujita scale from 1973.

CHAPTER VII

DISTRIBUTED UAV SYSTEM DESIGN

The focus of this chapter is to describe a method to perform distributed aerial assessments of disaster areas to give first responders, and, eventually, researchers, a better understanding of the immediate impact of a tornadic storm. The author believes that applying DARTS to a distributed set of low-cost autonomous aircraft can provide safe, *rapid* response, canvassing more ground in a shorter period of time and more effectively than by human operators in a single aircraft. By employing multiple planes, together with the DARTS reallocation mechanism, fault-tolerant disaster area assessment can be performed with a low startup budget, and negligible operating costs. The following sections describe the current techniques and costs associated with aerial assessment, and demonstrate the use of DARTS with a system called the Distributed Apt Resource Transference for Broad-response Overhead Airborne Reconnaissance Dispatch (DARTBOARD). Simulated and real-world trials are used that will present an alternative to human assessment in a way that saves time, expense, and lives.

Traditional Manned Disaster Area Reconnaissance

With knowledge of atmospheric temperature gradients, temperature boundaries, moisture levels, and upper-level wind shear, meteorologists can, to some degree, predict when conditions are optimal for the formation of tornadoes. And, given the magnitude of these variables, forecasters can, to some extent, predict the possible severity of such storms. With newer Doppler RADAR technologies and weather models rendered by supercomputers, forecasters can even predict general locations that might be more

susceptible to tornado strikes. However, they can neither pinpoint the time or location of a tornado strike. For this reason, gathering information about storm aftermaths can be vital in improving future forecasts and severity predictions.

The focus of this chapter, and of the dissertation itself, is to use DARTS, applying it to the use of distributed autonomous aerial vehicles (UAVs) that can perform distributed aerial assessments of disaster areas to give first responders, and, eventually, researchers, a better understanding of the immediate impact of a storm. The justification for employing a robotic solution is simple. Currently, such observations are done using manned helicopter flights. This presents a number of obstacles and possibilities for error, which will be outlined in detail.

Current Equipment

The most commonly-used helicopter in use today, especially for use in aerial photography (including news casting) are turbine-powered aircraft that employ Brayton-cycle rotary internal combustion to drive a shaft at a high number of revolutions per minute (RPM). These engines are called turbo shaft engines, and are derivative from turbojets, but do not have the same exhaust pressure (instead directing most of the energy into the output shaft). Gas turbines, especially when driving a shaft, require significant quantities of fuel. As of the time of this writing, turbine-driven helicopters consume fuel at a rate of over 1000 USD per hour. With the addition of maintenance costs, pilot salary, aircraft and airport permits, and other necessary equipment, the cost of running aerial disaster area assessment can be prohibitively expensive, especially for local governments

that have just been hit by a tornado and are facing significant future infrastructure expenses to rebuild and rehabilitate a community.

In addition to operating expenses, the use of a helicopter introduces a single point of failure. Not only is the craft human-operated; any malfunction, equipment failure, or other problem could ground the helicopter, or worse, create a problem in flight with potentially catastrophic results for the helicopter and its crew, as well as those on the ground. From an operational standpoint, this single point of failure can be problematic, particularly when lives are on the line from a first-responder's point of view. Logically, more helicopters and crew could be added, but this only further complicates the survey (including logistical challenges of coordination) and adds tremendously to the cost of running the operation.

Designing DARTBOARD for a Candidate Storm Intensity

DARTBOARD should be designed to survey a disaster area typically found after a tornado. Based on the data on tornadoes from 1950-2001 collected by NOAA, the next logical step is to design a solution to provide superior coverage, speed, and safety; to do so, this dissertation has focused on a target storm category that weighs the frequency of storm types (based on the original Fujita scale), the risk to humans (referring to damage that can be imparted by the storms in both casualties and property damage), and the probability of each storm type's occurrence (derived from the frequency analysis). The results of this detailed analysis yield a target storm type of an F2 tornado based on its probability of occurrence and its human impact. Let us briefly analyze how this target determination was made.

Table 2 shows the probability of a tornado's severity in fractional values adding up to 1. The distribution is also shown in Figure 26.

Table 2 - Tornado Intensities and Probabilities

Fujita Scale Intensity Index	Probability of Occurrence
F0	0.28
F1	0.39
F2	0.24
F3	0.06
F4	0.02
F5	0.01

From this data, we can see that the most common types of storm occurrences are F1 tornado types – according to the 2001 report. Note that this statistic has changed; in more recent interpretations of the original 1973 Fujita scale, F0 to F5

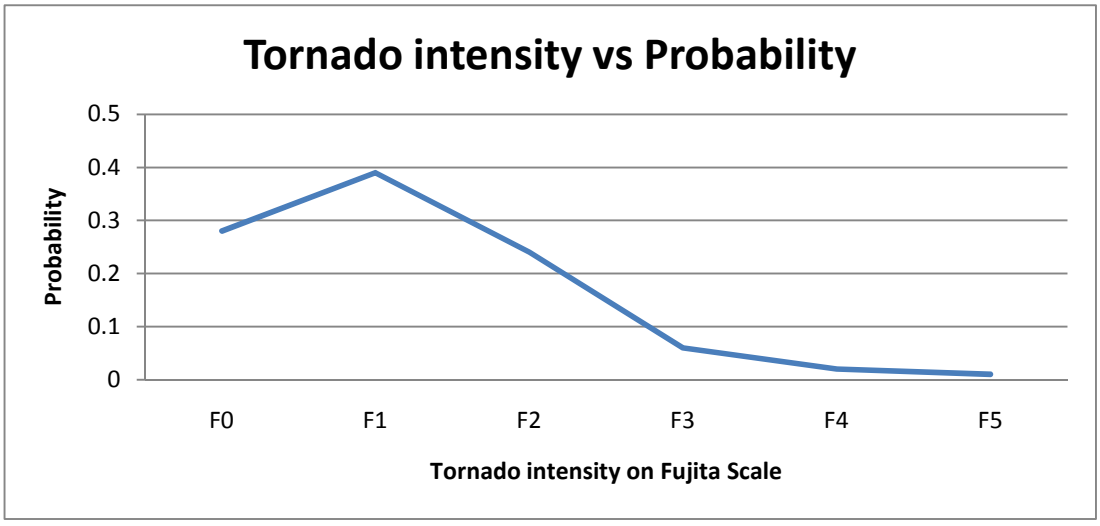

Figure 26 - Intensity vs. Probability

probabilities are listed as 0.39, 0.36, 0.19, 0.05, 0.01, and < 0.001. However, because this dissertation analyzes storm statistics in line with the data gathered in the 1950-2001 report, the frequencies in Table 2 will be used instead. A likely reason for the change in

probabilities is the increased resolution of weather RADAR, and improved communication between local storm spotters and storm predictions and response centers. Increased RADAR resolution can explain why more F0 storms are detected; while they may not produce damage on the ground, they can be seen in RADAR images at a certain detail threshold. Otherwise, the storm could conceivably be missed. Logically, because the likelihood of tornadogenesis are low when compared to the number of overall storms and severe storms covering the United States, it is reasonable to assume that many of those storms that do form will be low in intensity due to minimally-adequate updraft zones and insufficient thermal and moisture gradients to support larger storms; larger storms require significantly more atmospheric energy.

Prior to 1994, the mean width of a tornado's path was listed for each storm. Since 1994, new data has been recorded in terms of maximum storm path width. [87] Logically, stronger storms with well-defined updraft zones and sufficient atmospheric energy on which to feed tend to have longer storm tracks, as they last longer. In order to design a system based on an "ideal" target tornado, storms with either low probabilities or low risk of damage were eliminated from the set of tornadoes from F0-F5. A subset, spanning storm intensities from F1-F3 was therefore selected because F0 storms typically produce little or no damage, and the likelihood of an F4 or F5 storm is less than 2% according to the 2001 NOAA report. Though F4 and F5 storms can cause catastrophic damage, designing a system exclusively around these storms may be impractical due to their low frequency, especially for locations where tornadoes occur less often.

Once the subset of storms was selected, their probabilities were re-computed. The probability was computed assuming that all storms that occurred fell within the range, such that the subset's cumulative probability was 1. Table 3 shows these updated probabilities for the subset of storms F1 through F3. Figure 27 shows a plot of these probabilities.

Table 3 - Normalized Storm Probabilities

Fujita scale classification subset	Normalized probability
F1	0.565217
F2	0.347826
F3	0.086957

Based on these statistics, an F1 tornado is most probable. Of course, this fits identically to the findings of the superset of storms. Given the computed subset probabilities, a weighting scheme was needed in order to determine the impact of a storm in terms of casualties and property damage. To this end, the probabilities were re-computed by using weighting scalars on the array of F1-F3 storm occurrence probabilities. These values were chosen based on predicted human impact, but are **not**

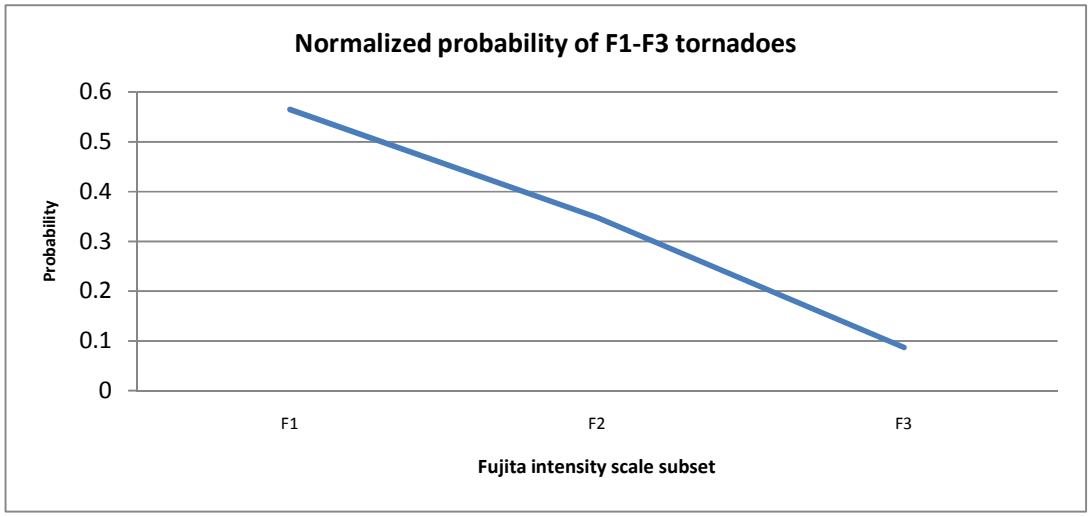

Figure 27 - Normalized intensity probabilities for the F1-F3 subset

scientifically derived as there are no absolute or average measurements of damage based on storm intensity types. Because F3 tornadoes can cause severe damage, their probability factors were multiplied by 0.7, F2 storms by 0.2, and F1 storms by 0.1. These scalars were selected because weighting F3 storms heavily due to their high amount of damage seems reasonable. The scores were then re-normalized and plotted to yield which storm type should receive primary focus.

Figure 28 shows this plot, and shows a well-defined maximum (despite the large weighted

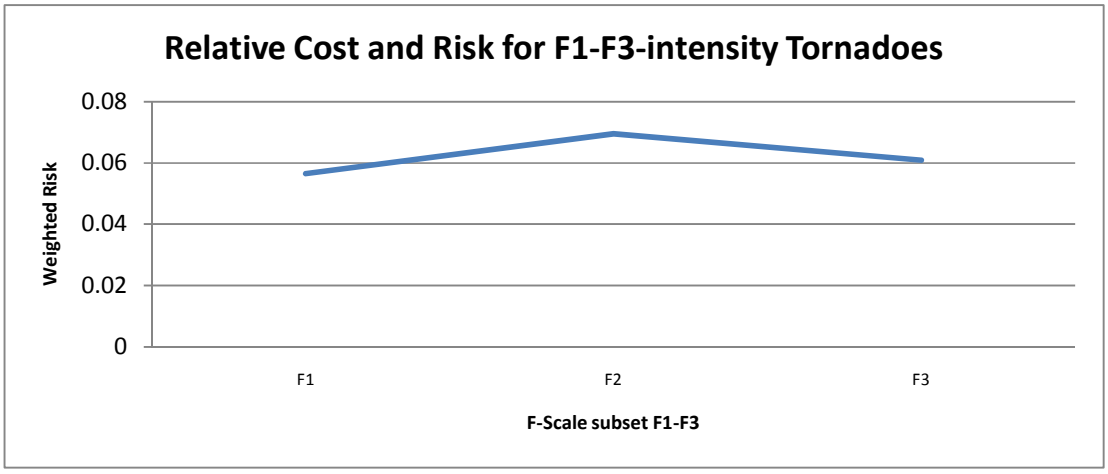

Figure 28 - Impact assessment normalization for F1-F3 storms

advantage of F3 storms) indicating that F2 storms represent a design constraint balance in terms of probability of occurrence and the weighted potential damage. Recent reports independently suggest that F2 storms are likely the most relevant of tornado phenomena [87], lending credibility to the subset selection and weighting model described above. This gives us a model we can now use to design a system within a certain set of specifications dictated by a representative storm, producing an aftermath that needs to be assessed efficiently and effectively. For this reason DARTBOARD is designed to

effectively assess a disaster area typically resulting from an F2 storm. Because the system's design and control constraints are actually quite flexible, the system is by no means limited to F2 storm damage; simply adding more nodes or re-specifying resolution requirements and setting altitude changes can easily account for larger or smaller storm tracks. In addition, because tornadoes vary greatly in track length, size, and path shape, the system is over -specified in order to account for natural variability.

Configuring an Airframe

Designing an airborne distributed sensing application requires thought in selecting an appropriate airframe to be used by each sensing nodes. An airframe, meaning the structural and propulsion aspects of an aircraft, can dictate almost all the performance characteristics of an airplane; wing design (mounted high or mounted low, or biplane), camber (the degree of curvedness of the top surface of the wing), fuselage length (long tail designs lend more stability), wingspan (determines loading), shape (whether the plane is a delta-wing aircraft or a standard dihedral configuration), and construction material, such as wood, carbon fiber, or foam. In addition, the power plant (electric, gas, turbine, or electric ducted fan) is critical in providing a good tradeoff between takeoff speed and distance, long flight times, and sufficient power to compensate for winds, drag due to heavier weight, steep climbs, emergency maneuvers, and other power-related factors. Because the target application for DARTBOARD requires a light, mobile, and cost-effective aircraft solution, full-scale aircraft are not considered; instead, model radio-controlled (R/C) aircraft make a perfect fit for short-range overhead reconnaissance and assessment. Because of their agility and low cost, these aircraft can be implemented in

almost any environment with restricted takeoff and landing conditions, reducing cost by their simplicity and materials.

In particular, this dissertation implements DARTBOARD on airplanes, rather than helicopters or Micro Aerial Vehicles (MAVs), such as quadrotor aircraft. Because agile helicopters employ collective pitch rotor heads, their complexity, maintenance requirements, and high cost preclude them from use. In addition, because keeping a helicopter airborne requires large amounts of power, poor battery life or high fuel consumption rates typical of radio-controlled helicopters make their use impractical. However, helicopters do feature unmatched maneuverability and the ability to hover in place; to solve some problems of helicopters (in particular their difficulty of flight), there exist quad and six-rotor aircraft that utilize distributed electric motors to provide cyclic flight response and hovering capabilities. However, even in their most basic configurations, these aircraft are expensive, exceeding $15,000 USD as of the time of this writing. Because such aircraft do not have collective pitch, rudders, or other moving control surfaces, a computer must be employed in order to allow a human to operate the device with controls that might be familiar to an R/C operator. In addition, helicopter-like aircraft have limited means of dealing with propulsion failure; a well-designed airplane can easily glide back to its runway without risking damage and injury to itself, others, and its operator. A helicopter at altitude that loses power must perform an autorotation maneuver; this can only be done once (because the rotor blades will slow due to drag) and must be enacted immediately to prevent a catastrophic event. MAVs, such as quad-rotors, are exclusively dependent on their power source to operate individual propellers; unlike a collective-pitch helicopter, where the blade pitch can still be modified in case of

engine failure to allow for autorotation and an attempt at landing, MAVs do not have any such capability. If one or more motors fail, a MAV will always have a catastrophic landing.

When considering R/C aircraft, there exist two main construction materials; foam and wood. Although electric foam aircraft are gaining popularity due to their low weight and inexpensive cost, they do not presently offer sufficient practical flexibility to load heavy sensors and communications equipment. Therefore, due to cost, design, and operational reasons, this work has selected the use of large wooden trainer R/C aircraft, which offer slow and stable flight, high payload capabilities, resilience to wind conditions, and ease of flight at the controls. These planes feature wingspans exceeding 1.6 meters and can carry over 1 kg of payload without difficulties. This dissertation names two candidates (one has been implemented) that fit desirable weight, flight time, and flight characteristic requirements. The first, which is functional as of the time of this writing, is the Hangar 9 Alpha 40, developed by Horizon Hobbies. Figure 29 shows a stock Alpha 40. A “40-size” trainer refers to the cubic-inch capacity of the nitromethane-fueled engine that typically power these aircraft; most engines have a volume of 0.4 cubic inches. Also called glow engines, nitro engines use glow plugs and compression ignition to provide power to a single front-mounted propeller. As the nitromethane fuel (consisting of 15% nitromethane, 80% or higher of castor oil, and a small quantity of methanol) is compressed in the 2-cycle, single-cylinder engine, the methane reacts with the platinum

Figure 29 - A stock Alpha 40

glow plug at high pressures and causes it to glow right as compression reaches its peak. Ignition occurs, and the cycle continues. There exist 4-cycle variants, but these are not price-competitive.

The rest of the airframe is simple; it features fully-functional control surfaces, such as ailerons, elevators, a rudder, a steerable nose wheel, and throttle control, and has a detachable wing for portability; the wing is fastened using rubber bands that are easily removable, and flexible in the case of a hard landing. It employs tricycle landing gear (with a nose wheel in front) with flexible landing gear struts to cushion harder landings. Using a 2.4 GHz DSSS uplink from a transmitter, the aircraft features a multiple-channel Pulse-Position Modulation (PPM) receiver that drives servos using Pulse-Width Modulation (PWM) signals. One or more servos are employed to move the control surfaces with control loops operating between 60 to 300 Hz, depending on the required application. The plane is balanced so that its center of gravity (CG) is less than a centimeter aft of the wing's chord line. Figure 30 shows where this center of gravity is on the Alpha. Coincidentally, the CG is also positioned so that the maximum payload weight lies directly beneath it, maintaining aircraft balance regardless of how much equipment is installed. All of the wooden components, consisting of the fuselage,

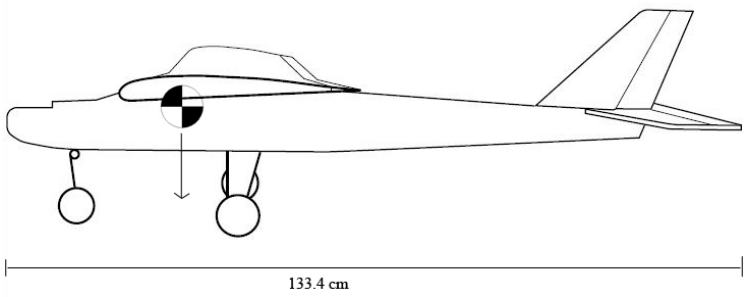

Figure 30 - The location of the CG on the Alpha 40

as well as the ribs and spars (the cross members that hold ribs in place, give the wing its length, and keep the wing mounted to the aircraft) on the wing, are coated with a heat-shrinking polymer (polyolefin) that adheres to the wood at one temperature, and then shrinks to tighten its fit at a higher temperature. This coating is necessary for the wings to provide lift, as well as to protect the aircraft's frame. In aircraft featuring internal combustion engines, this is especially important to prevent the degradation of wooden surfaces and components.

When configuring the airframe and selecting instrumentation, flight path, flight path, and flight conditions, three main factors of embedded systems physical design affect the aircraft's ability to fly, especially with small-scale-sized Unmanned Aerial Vehicles (UAVs.) These factors are:

- **Wing loading:** the capacity of an airframe to support and carry weight. Measured in weight per unit area, wing loading gives the capacity of a wing to provide lift. Note that the weight of the aircraft fuselage is *not* included in the wing loading quantity, and thus must be factored in when designing an airframe. Therefore, an embedded system must not exceed the wing loading capability, in order for the aircraft to be able to achieve lift. Further constraints are the physical load bearing of the wing spars. A wing may break due to momentary forces, even if the wing loading capacity is not

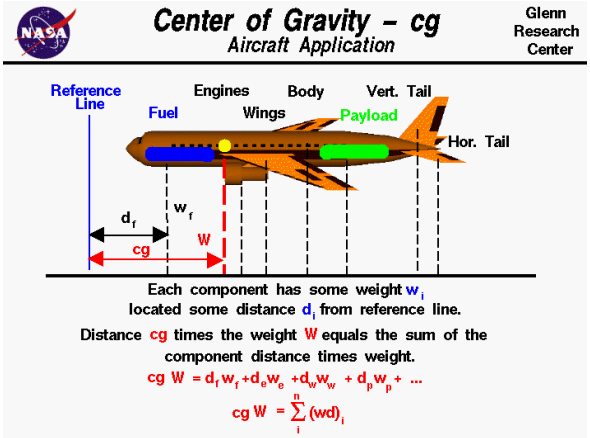

Figure 31 - The location of the CG on a generic aircraft (NASA GRC)

exceeded. A quick maneuver while heavily loaded is sufficient to cause catastrophic failure of a wing spar due to incurred G-forces and negative acceleration, resulting in the wings folding in on themselves.

- **Center of gravity (CG)**: The CG of an aircraft allows it to take *controlled* flight. Figure 31 shows an aircraft with a properly-adjusted center of gravity. Ideally, with a high-wing aircraft, such as the one used in this dissertation research, the CG is located just aft of the wing's leading-edge spar. If mounted too far forward, an aircraft will tend to tip forward; too far aft, and the plane becomes tail-heavy and difficult to control, especially at higher speeds. If the CG is not centered properly along the center line going through the roll axis of the aircraft, the operator or autopilot will need to compensate by applying ailerons to oppose the force (if so equipped). Thus, positioning of the system into an airframe is critical, unless the aircraft is large enough where small weight offsets will not adversely affect performance.
- **Drag** is a big factor that affects operational efficiency. If an aircraft is loaded with mass that is below wing loading, but representing a significant portion of the total payload capacity, the aircraft will tend to sink, requiring more power from the propulsion system. In UAVs, requiring more energy can lead to incomplete surveys, as battery life is also a scare resource. Gas or nitromethane-powered aircraft will also consume more fuel, cutting short the usable timeframe of the aircraft.

Additionally, onboard systems typically may require their own power source. Because high-torque electric motors may generate feedback into the power system

(unless equipped with an opto-isolated speed controller), undesirable power characteristics may occur when operating sensitive equipment. Furthermore, if an aircraft is electrically powered, additional drain on the principal battery is undesirable, and battery exhaustion (even with aircraft equipped with low-voltage cutoffs) can cause serious problems for onboard electronics and radio communications systems. Aircraft batteries are considered exhausted when they are unable to provide power to the motor or Electronic Speed Controller (ESC); they are still able to power servos and receivers in some cases, in order to safely bring the aircraft down by gliding. Further discharging a lithium-ion polymer battery is then considered over-discharging. Therefore, the addition of an extra avionics battery must be factored in to the system design, and therefore in the overall weight profile. The Alpha is equipped with a primary flight battery to power the motor, and a second, isolated, and smaller battery to power its computer systems, receivers, and servomotors. Mounting avionics, sensors, and other embedded systems will have consequences that can dramatically affect airworthiness of an unmanned aerial vehicle distributed sensing platform.

The Alpha implemented for this work has recently been converted to an all-electric platform. Substituting the heavy 2-stroke nitro engine for a 12-pole, 16-stator brushless motor capable of consuming 800 watts of power, the plane is now equipped with considerably more power, if needed. An electric conversion offers significant benefits to nitro or gasoline engines, especially when managing a large fleet of distributed aircraft. First, electric motor equivalents can provide much more torque, and better flight characteristics. They also do not suffer from flameouts, which occurs when a poorly-tuned or malfunctioning engine’s combustion cycle is interrupted. This can lead to

catastrophic failure, as R/C aircraft generally are not equipped with ways to restart the engine. Secondly, an electric motor is quieter and cleaner, and has no emissions; 2-stroke nitromethane engines are famous for releasing a slurry of un-burned castor oil, which serves as the engine's lubricant, out of the muffler and down along the aircraft's side, requiring solvents and a cleanup procedure following each flight. Despite the battery's weight (electric R/C aircraft use Lithium-Ion Polymer batteries), electric aircraft have a far better power-to-weight ratio, and are lighter, allowing the airframe to be loaded with additional equipment. Lastly, although the batteries themselves have significant hazards when shorted or charged improperly, electric planes do not need fuel that is inflammable, reducing on-the-ground safety concerns when fueling and maintaining the aircraft.

The Alpha is equipped with an ESC that is capable of supplying up to 60 amperes of current, at a voltage depending on the particular battery pack used. Batteries are typically measured in terms of their series rating; a 1S pack implies a single-celled lithium-ion polymer battery at 3.7 volts (all Li-Poly batteries have the same per-cell nominal voltage) whereas a 4S battery supplies 14.8 volts (4 1S cells in series). The capacity of a battery is rated in mAh. Each battery cell must be precision-charged by a balance charging system that ensures that no one cell exceeds 4.20 volts during charging; any overcharging causes intense heating, swelling of the battery, and often, fire. Li-Poly technology also is capable of supplying tremendous current; many batteries can continuously discharge at 20-40 times their rated capacity per unit time, without causing a safety concern. For instance, a 14.8 V (4S) 3200 mAh battery with a 30C where C is Capacity/hours rating can discharge at 30*(3.2 Ah)/h = 96 amperes. Many batteries also have a maximum burst discharge rate of up to 60C, which can provide almost 200

amperes of current. For this reason, caution must be taken never to short the battery terminals; a split-second mistake can damage or destroy the battery pack and cause a fire. Despite these risks, when charged properly and cared for, batteries provide a safe, and more reliable alternative to small internal combustion engines. Figure 32 shows the Alpha 40 in its new electric configuration. With a nitro engine, the Alpha 40 is capable of around 30 minutes of flight, possibly longer if the engine is run at a lower speed. Under electric power, the modified Alpha 40 can run up to 25 minutes on a single battery pack, provided light winds and efficient throttle management. The aircraft is able to operate continuously at speeds exceeding 15 meters per second (~35 MPH) and can exceed 27 meters per second (~60 MPH) with little effort. Glide and landing speeds can be as low as 3 meters per second in appropriate conditions (~7 MPH).

Figure 32 - The upgraded, electric power modification on the Alpha 40

An alternative (and complement) to the Alpha 40 is the Telemaster 40, from Hobby Lobby International (the radio-controlled division). This aircraft features a longer wingspan and slightly more robust fuselage design. Instead of a tricycle landing gear, it features conventional (i.e., tail-wheel) landing gear. Instead of a steerable nose wheel as found in the Alpha (which is controlled by the rudder input), the tail-wheel system's tail wheel is used for steering, and is attached to the rudder. A disadvantage of this configuration is that during landings, flaring, which is the practice of pitching up the nose just before

touchdown to reduce airspeed, is more difficult to accomplish. It does, however, require fewer servo linkages to accomplish rudder-based steering. Figure 33 shows a Telemaster 40 that has been converted from nitro to electric, as described above for the Alpha. This aircraft is intended for use as a second or third distributed node, but as of the time of this writing, has not been equipped with onboard navigation computers and hardware.

Figure 33 - A Telemaster 40 modified for electric use

Appendix A contains important information on the Alpha 40's autopilot system (Implemented with the Paparazzi open-source autopilot project) and its associated Ground Control Station (GCS). The reader is urged to complete this section to gain a better understanding of the Paparazzi autopilot system and its operating principles and specifications before covering the remaining material in this chapter.

Flight Planning

Based on the default configuration for a single-aircraft assessment setup, flight plans are made ahead of time (as mentioned previously) by creating an XML specification file containing waypoints and blocks. In the tests conducted for this research work, the waypoints were selected from a Google Maps overlay, representing key

coordinate points that resembled a necessary overflight of a disaster area. An average F2 tornado has a path length of 7.5 kilometers. For this reason DARTBOARD's flight plans need to incorporate this as an optimal path length. The tornado's actual path will be variable, twisting (pun intended) and turning in unpredictable ways. This path is based on the motion of its parent storm, as well as instabilities generated by atmospheric conditions immediately surrounding the tornado, which cause it to move haphazardly in some cases. Other storms, particularly wedge tornadoes, have a path that is dictated exclusively by its parent storm's updraft location. These tornadoes do not vary as much because of their structural soundness.

The 7.5 km path length only indicates an absolute overall storm path length. This does not, however, dictate that 7.5 km-worth of property and structures will be at risk. In addition, because a tornado does not maintain constant ground contact, the 7.5 km path length may be comprised of "spotty" coverage, where distinct sections were affected and others not. Because missiles can be generated and hurled several meters beyond the storm's funnel width, a buffer zone of 25 meters on each side is specified, yielding an assessment width of 175 meters. At a 7.5 km track length, assuming 5 aircraft and a funnel width of 175 meters, an area of 1.32 km^2 must be assessed. The Alpha can fly at a speed of 15 meters per second for up to 30 minutes under ideal conditions (a more realistic number is 25-27 minutes, which includes a high-speed and high-powered takeoff, and enough power to bring the craft in for a safe landing). At this rate, it can traverse a full 7.5 km path length in approximately 8 minutes. If necessary, this can be expanded – the Alpha can easily glide to a safe landing without power (known in R/C

hobbyist terms as a “dead-stick” landing, referring to engine and propeller stop), provided it has enough receiver/servomotor power.

However, this large of a damage area is outside the path length design constraint; local government agencies will typically see damage to their communities spanning a shorter distance; though the tornado may stay on the ground for a full average path length, the damage may not all occur in one community. For this research, a pair of housing developments in West Nashville, TN, spanning a curved path length of 1.4 km was selected. Although every community will vary, the selection was picked because of its proximity to the R/C flight field, in order to maintain legal operation guidelines with the FAA. Samples of aerial images from these developments can be seen in Figure 34. Large urban developments are not always favorable to tornado development, because large buildings

Figure 34 - Sample aerial images

disrupt airflow and provide uneven heating patterns which can affect or impede tornadogenesis. Cities emit considerable heat from the sun, but not all surfaces are identically colored. Of course, tornadoes do regularly strike cities, and therefore should be considered equally in this context.

Under a more automated scenario, a field operator would be able to select start and end-points of a disaster area, based on observed storm tracks and RADAR-based storm reports. If needed, intermediary points can also be added using a simple point-and-click-style GUI. By reducing the amount of time needed to assemble waypoints, an automated procedure could save lives and bring back data earlier. At present, the GCS does exhibit point-and-click functionality, but blocks must still be defined in the XML specification. Because Paparazzi and the GCS are free and open-source software (FOSS), the GCS is an excellent candidate for eventual modification to add point-and-click storm track targeting functionality. In addition, the GCS can overlay not only GIS information, but also custom imagery, such as observed storm tracks, to aid in the facility of deployment.

This dissertation has designed path selection in terms of its length (experimentally a 1.4 km swath.) While length is important, it is an easily-adjusted variable. A variable more difficult to identify is necessary flight altitude. A number of concerns govern an acceptable flight altitude. The first is obvious – proximity to the ground must be reduced to avoid interference with tall obstacles, such as antenna towers, multi-story buildings, and tree lines. Second, although most tornadic storms are followed by clear skies, this is not always the case. In this instance, flying at too high of an altitude could cause the

aircraft to enter cloud formations, obscuring any camera images. The third factor is camera resolution. The Alpha 40 test plane used in this dissertation was retrofitted with an Oregon Scientific ATC3K camera. Water resistant up to 5 meters depth, and virtually shock-proof, this camera is inexpensive and provided reliable video imagery for the test runs. Though the completed DARTBOARD implementation requires cameras that can upload data to a wireless link, or at the very least support a serial data interface, the ATC3K provided all the necessary imagery to conclude feasibility studies. The author custom-built a mounting solution that allowed the camera to be clipped to the side of the airframe before attaching the wing assembly. The camera itself was detachable by a pressure clip from the mount, for easy removal without necessitating wing removal.

Image resolution, in the case of the fixed-focus and fixed-focal-length ATC3K must be determined by the UAV's flight altitude – literally zooming in and out of a picture by changing altitude. Although higher-resolution, high-definition cameras with pan/tilt/zoom (PTZ) now exist, the 640x480-pixel ATC3K provided a good balance between resolution, cost, and eventual required data bandwidth. The aircraft was flown at altitudes between 31 and 731 meters off the ground (relative altitude, rather than altitude above mean sea level (MSL)). After experimental trial and error, it was determined that an altitude of 142 meters off the ground provided sufficient resolution in pixels per meter. The visible wide-view from the camera presented images 75 meters across. The resolution in terms of pixels per meter is therefore 8.5, yielding significant detail about structural elements such as rooftops, vehicles, roads, and trees.

Of course, as other sensors are fitted to the aircraft, a compromise in altitude, the addition of further aircraft, or repeated passes must be considered. Also, a heterogeneous sensing scheme may be implemented. In such a sample configuration, the first three aircraft in the pass are fitted with camera modules, capturing images at fixed time intervals. The two planes directly behind and staggered are fitted with infrared (IR) scanning equipment to locate survivors that may be present near or underneath structures during a night-time pass – where rescuers may have difficulty locating people due to low lighting conditions. Near-IR cameras may also be used in low-light situations to capture images that cannot be seen with a visible-light camera. And, cameras with pan/tilt/zoom capabilities can be added to give an operator control of what he or she is seeing. Each addition must be weighed and tuned to work optimally in the presence of other sensing equipment, whether or not the sensors reside on the same airframe or not. For the purposes of this dissertation, calibration will be centered on the ATC3K.

Critical to successful PID control is error minimization with respect to the aircraft's intended track. Because performing aerial assessments require precision in order to photograph or record video of the intended target, error minimization must be done thoroughly. Because a GCS was not employed, a test flight was flown with a secondary, track-logging GPS onboard. As both GPS receivers were given plenty of time to calibrate, measurements between the two receivers was measured to within 5 meters. In order to determine path accuracy, an ideal raster path from two waypoints signifying the start and end of a disaster area was plotted. Data from the tracking GPS was used to compare actual traversed path to the ideal track path. Despite any possible adverse wind conditions, the desired vs. actual measured waypoints differed by less than 45 meters.

Given accuracy concerns (assigning a maximum error of 5 meters to each waypoint), the difference shrank to a possible 35 meters of error in tracks between the actual and intended final waypoints. Because the aircraft must perform a relatively wide turn around the waypoint, this error is to be expected. Even so, the difference in actual path trajectories was not measurable (when considering possible deviations due to GPS accuracy), as it was within the GPS accuracy limitation. Because Paparazzi's LEA-5H GPS provides a speedy update frequency, this is to be expected. Furthermore, the AP's 60 Hz control loop response allows it to issue servomotor commands faster than the servos themselves can respond – negating any bottlenecks from the AP to the servo controls themselves.

CHAPTER VIII

IMPLEMENTATION OF DARTBOARD WITH DISTRIBUTED AIRCRAFT

This chapter describes how DARTS is used with the Alpha 40 UAV airframe. It describes how fault-tolerant techniques can assist the system in its assessment passes, ensuring complete coverage of the assessment area. Here, a number of configurations are described that are used in simulation. Real-world flight data from a single-aircraft experimental configuration is used to generate data that drives an Omnet++ simulation environment.

Multiple-Aircraft Operations and Deployment

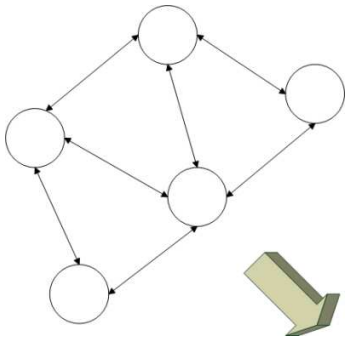

Figure 35 - The 5s configuration

Based on resolution information obtained from single-aircraft tests, three planes juxtaposed in a linear flight plan can then reliably cover an area of 225 meters, well beyond the 175 meter maximum debris range of an F2 tornado. Because of variations in flight path, this first line of aircraft would be followed by a second line of two juxtaposed aircraft that would superimpose between the two outermost tracks, in a staggered formation (the 5s configuration seen in Figure 35.) In the case of wider storm damage (caused for instance, by storms with widths between 200 m and 2 km or greater), either a different configuration will be necessary (the rear line in the staggered formation could be rearranged, extending a single-path standard-resolution pass up to 375 meters, or more aircraft could be added), multiple passes made, or a resolution compromise be reached.

An example requiring the extended survey area configuration is the Greensburg tornado of 2007. This storm was the first F5-category storm of the 2007 tornado season, and completely leveled the small town of Greensburg, Kansas. Figure 36 shows Greensburg prior to its destruction. As seen in the figure, the town itself was roughly circular in shape, with a few exceptions, and roughly 1 km wide by 2.3 km.

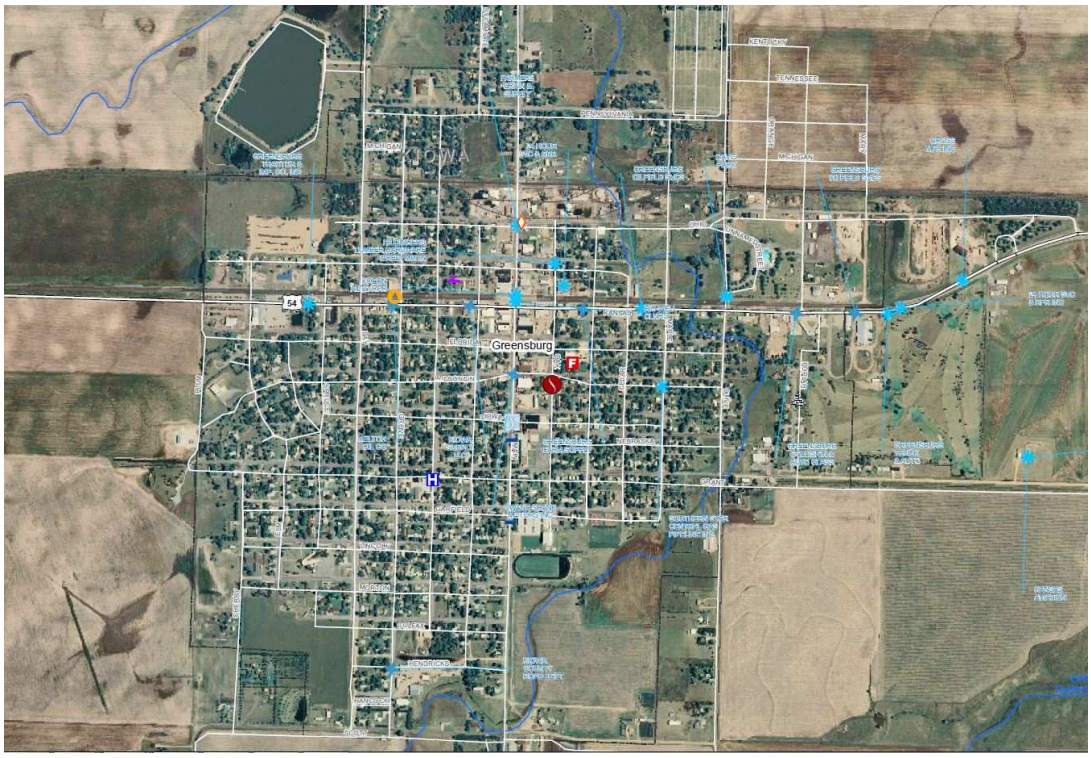

Figure 36 - Greensburg, KS, prior to the 2007 F5 tornado

The tornado that struck the city measured 2.7 km in diameter (significantly exceeding the town's width) and destroyed 95 percent of the town's buildings and structures. Figure 37 shows the city after the tornado's strike. In this case, the path of destruction exceeds the 225 meter assessment path width of the staggered aircraft configuration. This is the default configuration that will be referred to in this dissertation as configuration 5s – the first number indicates the number of aircraft, while the second

indicates the organization pattern – Staggered, Linear, or Normal (in which aircraft are arranged in squares, with all aircraft normal to each other)) is exceeded. These organization types fit clustering density categories of dense, sparse, and dense, respectively. In order to preserve linear resolution of 8.5 pixels per meter with the ATC3K camera, the 5s configuration would need to make four passes over the area. The 2.3-kilometer area can be covered in approximately 3 minutes, with an average aircraft speed of 56 km/h, corresponding to an average efficient cruise speed for the electrically-modified Alpha. This means that the overall time to rasterize the complete damage area would be 12 minutes, excluding the time it takes for the aircraft to reposition themselves.

Figure 37 - Greensburg, KS, follwing the 2007 F5 tornado

A number of factors will, of course, affect the overall flight time. First, a strong takeoff will deplete the battery somewhat; when simulating the amount of battery capacity that a full takeoff would require, this was experimentally evaluated to be around

200 mAh for full-throttle takeoff and climb of 60 seconds. In reality, a full-throttle takeoff is not required for the Alpha to become airborne, as the motor is far more potent than necessary; a full-throttle flight is highly inadvisable, as the aircraft would exceed speeds of 110 km/h and cause the wings to break due to stress. These speeds are significantly higher that what can be accomplished with the nitro engine, as it runs at a higher RPM but generates less torque, necessitating a propeller with reduced pitch; the overall effect is reduced thrust.

Factoring in the 200 mAh loss of a takeoff, the battery should last approximately 25 minutes of cruise flight per plane. In the extreme case, highlighted by the Greensburg storm, the planes would need to make each pass, and then travel a considerable distance before making the pass again. The return trip is divided into two legs, forming a triangle, as seen in Figure 38. As the planes approach the common point O (the origin), the planes *serialize,* meaning that they disassemble themselves from their 5s configuration and form a 5L configuration, seen in Figure 38. The reasoning behind this is twofold:

1.) Returning to a common point allows for fault-tolerant methods to take effect during this serialization point; if a re-scan is needed by one or more planes, these can continue being airborne, while the R/C pilot can land the remaining aircraft.
2.) When serializing, a common order is maintained, so that when the aircraft leave the common point, they can parallelize back into their native configuration easily. Furthermore, serialization allows the aircraft to each have similar path lengths, allowing for identical battery utilization during this particular leg of the operation. Ground controllers and pilots can also see the planes pass by and detect any

problem by looking at each plane as it passes, rather than having to search the sky to see each plane go forward.

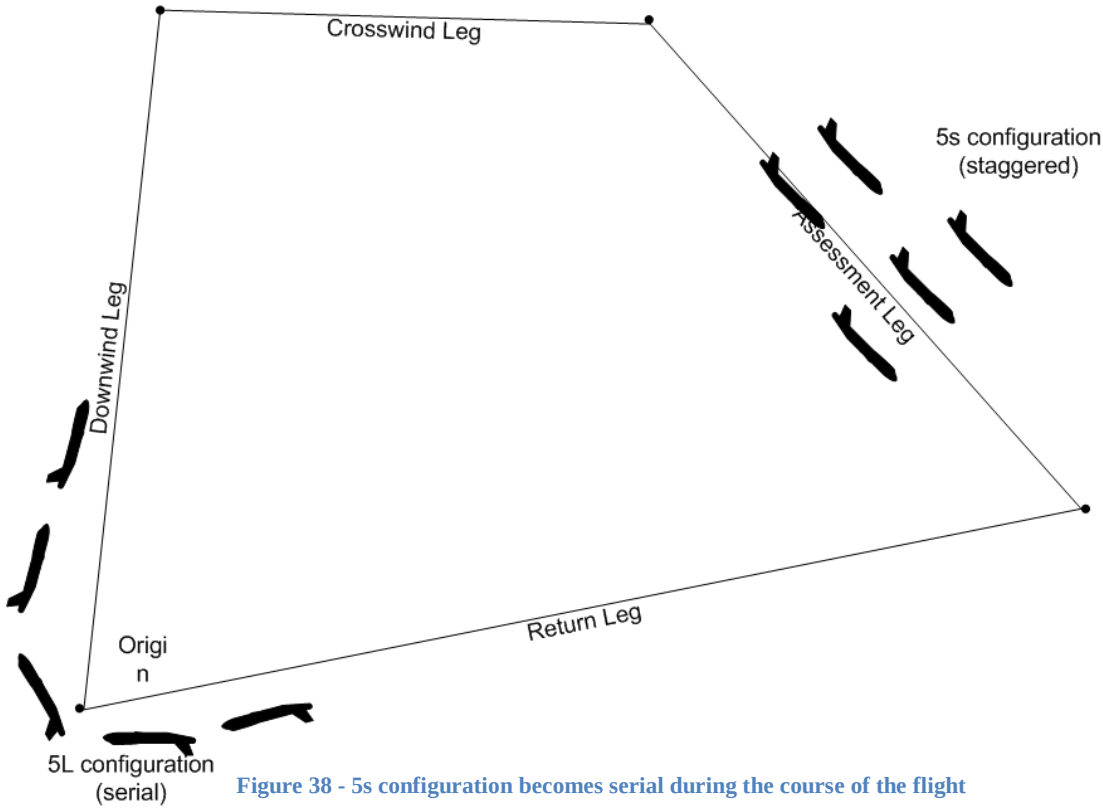

Figure 38 - 5s configuration becomes serial during the course of the flight

The lengths of the return trip legs, and the position of the serialization and parallelization points, are determined by a set of scalars. These were computed experimentally by analyzing flight video and determining the best way to allow smooth transitions between waypoints while simultaneously reducing the length of the return legs to reduce battery consumption. The point O was limited to the flight field's location. Based on experimental data gathered at the Warner Park flying field in Nashville, TN, lengths of legs to point O were set to three destination points: The first, the home point, in Figure 38, is the point where all aircraft return and should be in serial formation. The

distance from this home point O to the first turn was computed at 850 meters. This first leg, the downwind leg approaches the disaster area from the side and above in order to minimize course corrections from the second turn, after which the aircraft will start its assessment pass. The distance from the first point to the second point, called the crosswind-leg, was computed at 710 meters in length. The assessment pass is approached while exiting the crosswind leg at an obtuse angle; this is important because approaching a turn at a right angle can be a recipe for overcorrection by the autopilot's PID control. The final turn point, following the assessment leg is the return leg to the home point. By this measurement, the total flight path of the mean curved path was 2.8 kilometers. Flying at 56 km/h, the pass can be completed in approximately 3 minutes, plus or minus 12 seconds depending on whether or not a particular aircraft is following the center (mean) path or on the outside. The 12-second error is a boundary condition. Consuming 3 minutes out of a total of 27 "safe" minutes of battery capacity indicates that each aircraft is capable of performing either multiple passes, or providing redundancy in the case of a single or multiple point of failure. To understand how this can be accomplished, let us proceed to the next section where the implementation of DARTBOARD with the Paparazzi-based UAV will be discussed.

DARTBOARD Integration

When examining contents of Chapter IV, we remember that the reallocation of tasks is based on resource allocation of one or more resources assigned to a node. Therefore, we must answer the question of what in the disaster area assessment scenario is a resource, and what is a node containing those resources. The breakdown of resources to nodes is not as transparent as the generic case for DARTS, so it will be explained in

more detail. Specifically, there is a dichotomy between what resources *are* and where they exist and to whom they belong.

The first resource of interest is a radio link. Specifically, the MANET-to-GCS link, which must be capable of transmitting two-way information over a distance of 4 km or more, depending on the search path. The 900 MHz GCS link mentioned earlier is based on a MaxStream 802.15.4 modem pair with a power output capable of exceeding 4 km if properly powered. Power at the ground control station is flexible, because larger batteries or portable power inverters may be present. However, the aircraft, especially in the 40-size category, must be frugal in its weight considerations in order to achieve efficient flight characteristics, and to be able to carry any necessary equipment to complete its objective. A separate battery will likely be needed to power the transmitter to prevent back-EMF, which is an electromagnetic field that is induced in the three-phase motor power lines due to rotation from pole to pole. It is used for understanding the speed of the motor, but can also induce unwanted electrical noise into common power systems. Therefore, rather than equipping each aircraft with a plane-to-ground link, it is likely a better idea for efficiency and cost to equip two planes with the long-distance link. One of these links will remain active, while the other remains on standby; this eliminates interference (even though the 802.15.4 protocol specifies Direct Sequence Spread Spectrum transmission technology, allowing for the superposition of multiple signals in the same band space without data loss) and makes for a simpler ground control system. All other aircraft can use local, low-powered networks, such as Bluetooth or low-powered 802.15.4 variants. Because each aircraft will eventually need to return images (ideally in real time) back to the GCS and observers on the ground running the

assessment, data is routed through mesh protocols from one aircraft to the next, until it reaches the ground relay. The result of this arrangement is that we now have our first resource: the GCS link. Because two aircraft will carry the capability to link to ground (using paired transceivers that can hot-swap to the GCS), reallocation in this case means that the failure of either the entire aircraft or associated power systems, or of the radio link itself, will trigger the reallocation as one airplane or another will eventually request a link back to the ground to route data that it has gathered. During the time that no links are available, aircraft will need to cache their data until a link is re-established by the DARTS protocol. Because the ground link is monitored only by the aircraft on which it is implemented (neighboring nodes requesting data cannot and do not know the state of the GCS link), there is no good way to issue triggering based solely on the observation of neighboring nodes; unlike HybrIDS and related possible triggering methods, that rely on behavioral analysis at the application scope, the ground link is negotiated exclusively by its host aircraft. For this reason, based on principles implemented in DARTBOARD, the GCS link must fail first before reallocation can take place; timeouts associated with requests from neighboring aircraft will bring link failure to their attention, spawning the reallocation process. Because the timeouts will generally not occur until GCS link bandwidth has dropped to levels so low that it is unusable, or has been terminated altogether, pre-emptive reallocation is generally not possible. Were DARTBOARD to be modified to include regular status updates, or an advanced triggering method that proactively warns other nodes of link failure, this would be possible. However, because trusting information from a compromised node is always a security risk, this is not implemented on the version of DARTBOARD developed for this dissertation.

The second resource that we can identify, at least within the scope of disaster area assessment, is related to path management. Instead of focusing on a node's individual camera as a resource, we can also suppose that the path, which must be rasterized by a particular aircraft, is a resource that is *assigned* to an aircraft. As each aircraft nears its target run, its intended path is the optimal path based on its position. For instance, as seen in Figure 39, Node 0 is the most fit for Path 0, and Node 4 is most fit for Path 4. However, Node 1 is reasonably capable of assuming the path for Node 3 if there were a disruption; as there are five distinct paths, we can assign fitness scores to these methods of path management based on proximity of the aircraft in question. It is possible to reduce the number of fitness scores, as there will exist an overlap of used scores, as some aircraft are centrally-located and equidistant to other points. However, for reasons of simplicity, the fitness score will be identified by the total number of identical resources, which in this case corresponds to the number of available paths.

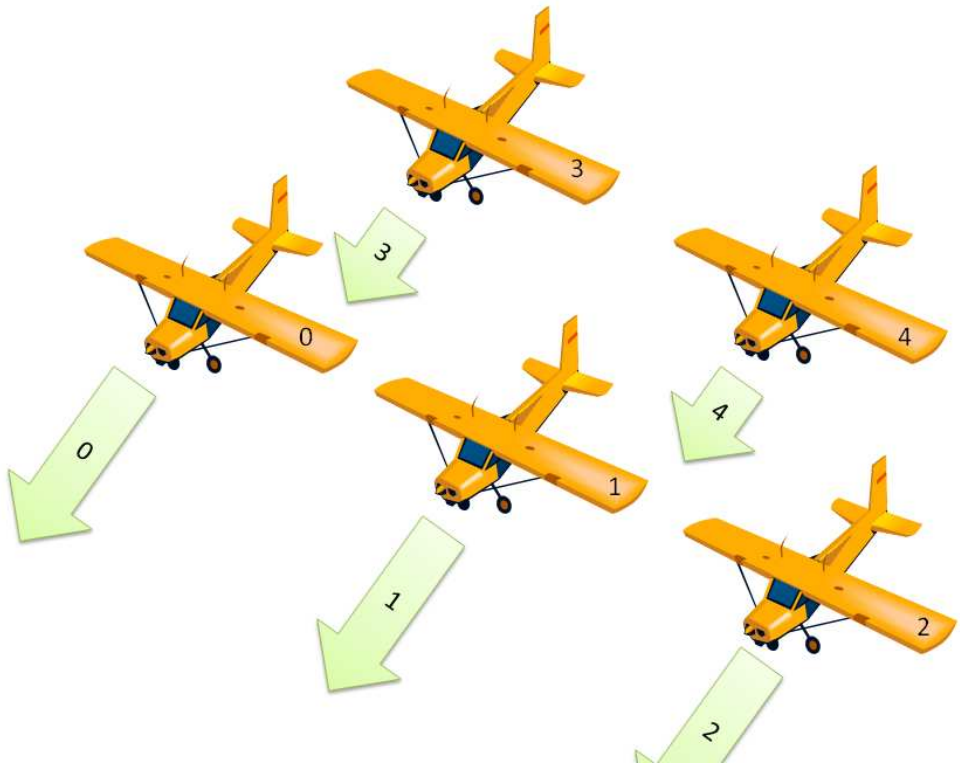

Figure 39 - Nodes and most-fit paths

Because it would be disruptive to have an aircraft change its raster path in mid-course as the result of a reallocation attempt, reallocation will have the immediate effect of defining a successor to the missing node's path. As the aircraft approach their point of origin and serialize, aircraft not needed as the result of the reallocation request can land, while the newly-assigned replacement node can complete the sweep that was left un-finished by the failed or compromised aircraft. The reader may at this point wonder why fitness scores are needed; any node can act as a replacement for a compromised aircraft, and at the serialization point, an arbitrary aircraft could be chosen to go back and complete the unfinished assessment. The answer to this is two-fold. The first, and the most obvious, is the case in which multiple passes are required to finish an assessment of

a large damage area. In this case, upon serialization, all aircraft will need to return to the area and proceed to scan again, at a certain offset. Meanwhile, the chosen replacement resource will return to the first scan's unfinished path and run the needed scan. At this point, the remaining group and one or more replacement resources are partitioned into two different areas. Once the primary group completes their scan, they can land after having serialized. The one or more aircraft finishing up incomplete scans can then perform their remaining pass(es) and land afterwards. This gives the system the flexibility to schedule passes on-demand according to most desired capabilities. If the replacement node to also fail during the re-scan of the first section, then a node on the second section could be reallocated, and it would be sent back to the start where the second node had failed to complete the pass of the first-compromised node. This can happen flexibly until all resources are exhausted. Using fitness scores with this scheme also permits conservation of battery life, by not arbitrarily assigning scans to resources that may already have performed extra scans.

The second benefit of fitness scoring is the implementation of heterogeneous hardware; if the two rear nodes in a sample 5s configuration were set to run infrared scans instead of using visible-light cameras, then the two rear-most aircraft cannot supply fit resources to replace nodes using only visible cameras. A single node could also be retrofitted with both resources, reducing cost and then be the most fit to replace either resource upon compromise. This permits the systems to partition node replacement based on individual resource capabilities. As such, these two benefits to using fitness scoring allows DARTS to be integrated directly, without modification directly into use in the disaster area assessment application.

DARTBOARD Triggering

When considering ways of triggering reallocation in the DARTBOARD implementation, there are two types of resource losses that occur with respect to neighboring nodes. The first is request-detected resource loss, in which a neighboring node can identify the failure of another node because it needs the other node in order to complete a task; there is a product-to-consumer relationship between the detecting node and the failing node. These cases are easily detected by an intrusion detection mechanism; when a resource becomes unavailable, or times out, a fault is suspected. (Of course, more elegant variations on this theme are likely, such as monitoring request difficulties before a complete failure occurs. Also, IDSs are able to pick up on abnormal behavior before a problem arises.) An example of this product-consumer triggering mechanism is the aircraft-to-GCS link that is present in one aircraft; if one of the distributed nodes attempts ground communication and difficulties arise, the failure is identified due to the dependency on the failed resource.

The second triggering scheme occurs when a node identifies a failing resource independently from whether or not the node needs the resource; for instance, a fringe node can fail, which is not used in data routing among the group. Although no other node will explicitly need resources from this failed node, they will notice a communications dropout. In this particular case, no other node's operation is threatened, but the overall objective is at risk and thus reallocation must occur. In our 5s example, an outside, fringe node may stop reporting images and status updates to its neighbors.

Before we can discover how DARTBOARD implements its IDS-based triggering approach, let us first discuss the behavioral interactions that are present on the MANET during normal operations. This gives a perspective on how an IDS should characterize system occurrences to determine whether or not reallocation triggering is necessary. Because HybrIDS technology is applicable to detecting whether or not hostile interactions are taking place as well as simultaneously being applicable to detecting abnormal interference in normal operations due to natural phenomena or system failure, we can extract useful techniques from it to apply to DARTBOARD. The HybrIDS method itself is a generic solution, and in this case, can stand modification to apply it more appropriately to a networked distributed UAV scenario. In essence, the implementation incorporates a "HybrIDS light", representing a useful set of IDS capabilities that are tailored to operation on DARTBOARD.

Updating Caches

A feature of interest to general and commercial aviation is the ability to broadcast identifiers that help prevent aerial collisions. There exist two primary methods used in aviation today: TCAS, and ADS-B. The Traffic Collision Avoidance System (TCAS) is a simple transponder-based location awareness broadcast system that aircraft can use to implement rudimentary collision avoidance. TCAS-equipped aircraft issue a general broadcast that polls for nearby aircraft. These aircraft then reply with their current position information so that the requesting plane can compute likely paths to prevent a collision. TCAS was first implemented in the early 1990s on all commercial aircraft (turbine-powered) with more than 30 passengers [44, 45]. Its evolution is the Automatic Dependent Surveillance-Broadcast system, ADS-B. ADS-B relies on the same principles

of operation as TCAS, but adds additional functionality and data, along with ground-based stations that can update the aircraft with information about: (1) aircraft not in range, (2) specific flight conditions, and (3) airport and runway conditions. The system is also extensible to provide cockpit displays and automated collision avoidance and airspace conflict management. The ADS-B system entered commercial use in 2006 and is likely to replace TCAS in the near future as the dominant collision avoidance and traffic conflict management system operating on aircraft, supplementing ground-based radar stations in their efforts to keep aircraft from colliding with each other.

Based on the principles of ADS-B and TCAS, DARTBOARD is outfitted with a rudimentary hybrid version of TCAS/ADS-B. Instead of having aircraft submit requests for position information and then wait for a reply, TCAS-like messages are broadcast from each aircraft in the DARTBOARD airborne configuration at a rate of 4 Hz. This behavior can be tracked by IDS components of neighboring nodes to determine whether or not an aircraft is compromised by interference. If a TCAS message is not received within a certain period of time, its neighbors know that a failure has occurred. Referring to the two types of triggering methods, identifying resource loss due to TCAS latency is considered a resource-independent triggering method; neighbors may not need the aircraft (other than for routing purposes, which can trigger the IDS on its own), but they can report its absence.

The TCAS message itself also contains a small table of resource fitness information, feasible at the small TCAS message level due to the small number of resources in this implementation – so that neighboring nodes can update their caches

opportunistically based on the TCAS broadcast. It also includes other optional information that can be used to transmit data form aircraft to aircraft, and includes GPS position information, autopilot operation state (i.e., whether or not the aircraft is under manual or AP control), and battery state. The battery state can be used by neighbors to determine unusual modes of operation; if one aircraft exhibits excessive battery use, it could indicate a failing power cell. If two aircraft are exhibiting extreme battery usage, this may be an indication that either the group is experiencing adverse weather conditions (e.g., requiring more throttle input) or that the other planes are experiencing abnormal radio or control surface utilization, possibly indicating a network intrusion or other fault. Of course, battery depletion may be the result of an aircraft having implemented a second pass due to resource reallocation, which would be expected and ignored as an indicator. Lastly, and optionally, rather than routing image capture data through the network by normal means, TCAS messages could theoretically embed photographic information. This capability has not been implemented.

In addition to TCAS messages and periodic GCS information updates, image data must be transmitted to the ground; rather than depending on storing the images on the aircraft themselves (image storing is already done in case redundant GCS links all fail), the data is transmitted directly to the ground to reduce the time needed for a ground operative to wait on data. With each aircraft operating at a preset flight speed (around 15 meters per second), images can be taken at preset intervals, presenting some image overlap, but not too much so as to prevent excessive bandwidth usage. However, if aircraft face heavy headwinds, slowing their progress despite additional throttle input, or if they face strong tailwinds that cause the aircraft’s speed to exceed its preset despite

throttle input reduction, this preset speed must be altered accordingly. Because sufficient field test data is not available, all simulation work is done assuming a 15 meter/second flight speed. Each photograph at 640x480 (VGA) resolution consumes approximately 40 kilobytes of space with high-quality JPEG compression, and must be transmitted every 2 seconds in order to capture detail of the scene at the given speed with an overlap of 90 pixels per image (approximately 15 percent).

Local networking between aircraft is implemented with the 802.15.4 protocol on 2.4 GHz links, and is capable of reaching aircraft within 150 meters of each other. For security and cost reasons, local networking is not capable of interacting with the GCS, requiring any node wishing to transmit data to the ground to use the GCS link, operating on a similar protocol, but with greater range and different operating frequency (900 MHz). The local network has a maximum bandwidth of 250 kilobits per second per node to maximize battery performance. This means that the transmission of photographic material occupies a network overhead of 64% (320 kilobits twice every 2 seconds, yielding 160 kilobits per second average) during normal use. TCAS messages are 256 bits in length to accommodate up to 32 characters of ASCII data, the default encoding format of the message data itself, plus an additional 32 bits of packet header for reserved use. Figure 40 shows the formation of this message. Transmitted at 3 Hz, which can provide sufficient reaction time in the case of sudden aircraft maneuvers or wind gusts, the 256-bit packets consume 0.1% of the available bandwidth. Naturally, one would expect the need to transmit more data than just the data of the plane itself; because mesh

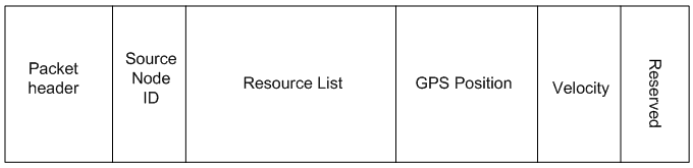

Figure 40 - TCAS packet organization

routing occurs on the network, a plane must be able to transmit more than just its own images on the data stream. This introduces a unique challenge not yet addressed by the 802.15.4 standard, which at 2.4 GHz does not provide more than 250 kilobits/second transfer speed. To circumvent this limitation, there are four possibilities. The first, and most obvious, is to choose a higher protocol, such as Bluetooth 2.0 Extended Data Rate, which supports throughput speeds of 768 kilobits per second. The second option is to reduce the speed of traversal so as to require a lower image frequency. Third, redundant radio links can be added to each aircraft, though this increases cost, weight, and power issues. The fourth, and likely most feasible, is to simply compress the images further. The 40 kilobyte image assumes a relatively high image quality. In fact, each image can be compressed to approximately 15 kilobytes (at VGA resolution) using the JPEG algorithm without significant loss in quality; though some loss is evident, it is not enough to impair visualization of the target area. At 15 kilobytes per image every two seconds, the mean data rate required is 60.3 kilobytes per second, allowing up to three aircraft to send their data through one node, reaching a total network utilization of 73%.

Figure 41 - TCAS packet organization

Because of the proposed 5s configuration, the data from no more than two nodes will ever be transmitted to the ground link at any time, with the exception of the serialization point configuration. Here, cameras are unneeded, so the problem is resolved.

By this point, we have discussed airframes, implementation, systems, resources (scan paths and GCS links), and local and ground communications. Let us now discuss how these systems were tested either in the field or in simulation to analyze the

performance of the described system. Because only one aircraft was autonomously-capable as of the time of this writing, in-field test were done to determine ideal scan paths, traversal speed, battery life, image resolution, required altitude, scan path width, and communications range, among many other variables. Therefore, trials with multiple aircraft were done in simulation to extend the findings of the single-aircraft field trials. Simulation was performed in Omnet++ version 4.0 [91], an open-source, widely-accepted and validated network simulator.

Because Omnet++ does not feature a native mobility simulation component, integrating actual flight paths from field tests would prove to be tricky. Although a third-party mobility framework, called MiXiM, is available, it is in early stages of development, and lacks in robustness and usability; with time, adding this component into Omnet++ could prove more fruitful. Instead, the target 5s configuration was broken down into a "basis set" of possible orthogonal configurations that mimic different stages of the flight path. For instance, the target of the pass, the 5s configuration, is maintained constant throughout the pass, and well into the serialization and parallelization points. Of course, as would be expected, the link lengths will vary constantly; however, because network links provide nearly-instantaneous communication between networked nodes, changes in distance do not reasonably affect the operation of the network in any way, provided that all aircraft stay in range of nodes in compliance with the 5s configuration.

There are five principal configurations that were implemented; the approach, the 5s assessment pass, the parallelization point, the serialization point, and the serial flight path over the ground control station (i.e., the origin point.) These are five unique

configurations; many non-orthogonal configurations exist, but are really duplicates or modifications of the five listed here. For instance, an identical network configuration will exist between the serialization point and parallelization point; for this reason, those two configurations are used not for their name but simply to outline two possible situations under those conditions. The intended configurations represent likely possibilities that the aircraft will take under normal operation. In practice, there may be some unaccounted configurations in this method, but they are not investigated in the dissertation. A more conclusive study involving a different simulation mechanism is expected to be conducted in the future.

The Omnet++ outputs that are most interesting to analyze are time to reallocation and the number of messages required to reallocate using DARTS versus utilizing the baseline flooding algorithm (similar to the study in Chapter V.) The network itself is capable of simulating the average network traffic (i.e., image data) plus the TCAS messages and mesh routing to reach the GCS. TCAS messages were configured to run at 3 Hz intervals to allow for sufficient time to correct for proximity errors; although the onboard GPS systems themselves update at only 2 Hz, having an asynchronous position monitoring scheme allows for a bit of overlap should a critical decision be necessary; setting a 2 Hz TCAS rate yields a higher probability that the GPS position updates and the TCAS broadcasts are out of sync by one message. The HybrIDS Light implementation is an insertable Simple C Module construct that integrates within Omnet++'s runtime to allow for two types of monitoring: The first is TCAS packet rate monitoring to check whether or not traffic from neighboring nodes is received at 3 Hz plus or minus 30%, in case of delay due to either a delayed I/O response or processing

overhead. The second is a link request timeout for the GCS link, based on a timeout value (500 ms to account for possible buffered traffic). Each of these two scenarios employs a specific implementation of the larger HybrIDS scheme; rather than using its complete, multiphase intrusion detection systems, condition checking is called for (and available as a feature in the original HybrIDS kernel). While it would be beneficial to include the full behavioral monitoring and IDS strategies of the multiphase solution to test for intrusion anomalies, it is beyond the scope of this work. It was mentioned earlier that monitoring radio link power level could be used to identify potential problems. Because there is no standardized power framework in Omnet++, this has not been implemented. However, the related idea of monitoring packet retry requests is being implemented in future versions of the mobile-aware DARTBOARD system.

In order to test possible error conditions, such errors must be identified preemptively so that they may be included in the simulation. Because the triggering scheme is designed to identify operational errors, most possible failures, such as link failure, hardware faults, software bugs, jamming, or radio frequency interference, are automatically included as possibilities in the simulation environment. However, other possibilities exist, such as a cache failure, cache staleness, fault identification failure, and combinations of errors. Most of these can be addressed because of the periodic nature of the system; Failures affecting performance and response time will almost always be identified by the triggering mechanism, causing reallocation to take place. Of course, by this reasoning, a simple and quickly-resolved error could result in a false-positive identification, or an unnecessary reallocation. From the point of view of the author, it is preferable to reallocate tasks to other resources, especially if small errors are the

precursor to a larger failure on a node, such as indicated by memory failure or processor cooling issues. Another potential failure could be due to power supply problems, such as batteries, power conditioning systems, and voltage regulators. As systems increase their complexity, small errors may be the only way to prevent a catastrophic failure from occurring later.

Caching errors represent a measurable fault that can be detected and tested in simulation. Because caches are updated with information from the TCAS broadcast at a rate of 3 Hz, it is highly unlikely that cache coherency and staleness would ever be a problem. Because each node is only aware of resources of neighboring nodes, other nodes more than one hop away are not concerned with these resources, and therefore do not need synchronization of their caches, which could create cache coherency problems between nodes separated by a certain number of hops. This is where DARTS has an advantage over a gossip-based protocol, because caches are updated frequently by the TCAS broadcast and do not, therefore, suffer coherency problems. Invalid cache references to a failed resource are impossible because each reallocation request contains a method of invalidating the cache entry associated with that resource. The only remaining staleness concern involves entries that are invalid because a neighboring resource has moved away from formation. This is also an improbable scenario, however, because at the rate of relative forward and lateral movement will never be fast enough to cause staleness concerns if the TCAS broadcasts are done at 3 Hz intervals; forming a new aircraft configuration from the 5s formation in under one third of a second is not physically possible. (Transitions to different configurations are stepwise – they occure instantaneously as aircraft move in and out of radio range.) This is especially true

because each aircraft is separated by 75 meters from side to side, and an additional 100 meters or more from front to rear. However, there does exist the possibility of a failed cache. A node could have a fitness cache is simply not operating for one reason or another. Therefore, DARTBOARD is tested in simulation with randomly-determined failed caches; a resource reallocation attempt is made with one or more nodes having a disabled cache in order to simulate this condition. In the 5s configuration, one or two nodes were randomly picked to have inactive caches, using the atmospheric-interference true random number generator (using radio interference values to generate true random numbers) API from random.org. With 5 nodes, cache failures occurring in more than 2 nodes would cause the system to approximate the baseline condition, depending on the node's location; this negates further testing. Recalling that during a reallocation attempt, four aircraft or less will be present during a 5s configuration, the selection of two failed nodes (50%) seems a reasonable compromise. Start position (i.e., the node running the reallocation, as well as the failing node) were also randomized.

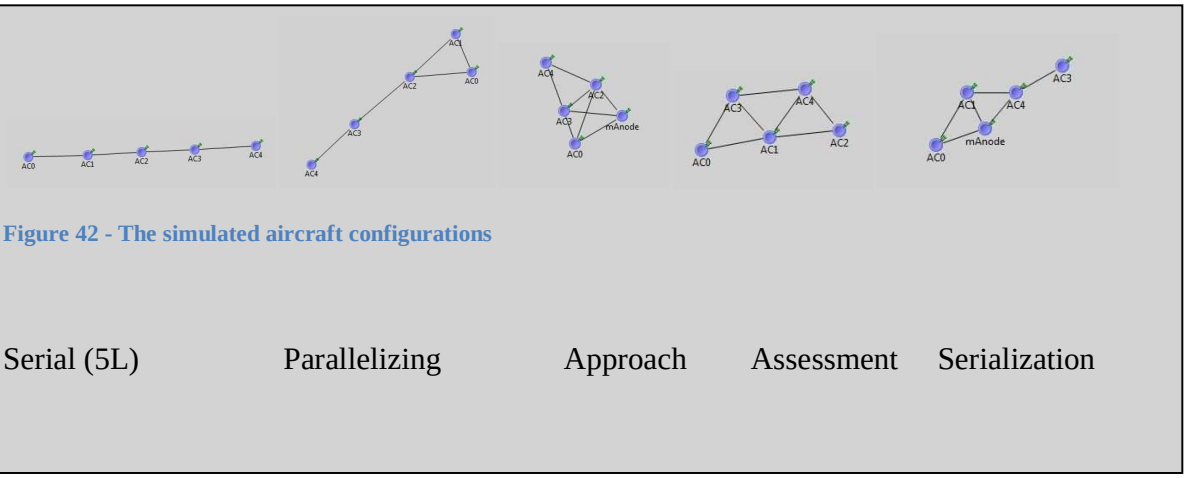

Figure 42 - The simulated aircraft configurations

Simulation Results

Simulations were run for the five configurations mentioned earlier. These are seen, without the removal of the failed nodes, in Figure 43. Table 4 lists each trial and

configuration, with baseline (non-cached) and cached results. Reallocation times (in seconds) and the number of required messages are listed, along with any pertinent observations that may serve to clarify the particular result. Table 5 shows the same trials run with one and two failed node respectively. While somewhat more difficult to visualize, the light-orange fields show that speedup and message improvements are still considerable. The randomly-chosen node is indicated in the configuration description column. The tables also show whether the reallocated resource was a radio link (GCS) or one of the five available paths in the 5s configuration.

Table 4- Variable-configuration simulation results

Configuration	Failed Node	Baseline Messages	Cached Messages	Baseline Time (s)	Cached Time-s	Message Improvement	Speedup
5s Approach	1	8	3	0.69	0.22	62.50	3.18
5s Assessment	3	6	2	0.67	0.10	66.67	6.58
5s Serialization	4	3	1	0.26	0.14	66.67	1.83
5s Serial	2	3	2	0.41	0.39	33.33	1.06
5s Parallelizing	1	6	2	0.34	0.14	66.67	2.50

Table 5 - Simulation results for single- and dual-node cache failures

Configuration	Disabled Nodes	Cached Messages	Cached Time (s)	Message Improvement	Speedup
Approach	1	3	0.11	62.50	6.43
Approach	0,1	3	0.11	62.50	6.27
Assessment	1	4	0.30	33.33	2.22
Assessment	1,4	5	0.27	16.67	2.47
Serialization	3	2	0.11	33.33	2.40
Serialization	1,2	2	0.30	33.33	0.88
Serial	4	2	0.36	33.33	1.16
Serial	1,2	2	0.34	33.33	1.23
Parallelization	1	4	0.11	33.33	3.18
Parallelization	3,4	3	0.11	62.50	6.43

Based on the data obtained by seeking a random resource from a random node failure, and with the additional test run including one or two random node failures, it becomes apparent that the caching algorithm presented in DARTS is an effective way to curb unneeded reallocation requests. Figures 44 and 45 show the percentage improvement and the speedup gained from using DARTS in the fully-functional cache model. Each reallocation request returned resources fairing a fitness score of 3 or higher on a scale from 0 to 5 (the score of 3 is chosen to select an above-average (50%) replacement resource). This automatically eliminates unneeded result returns, and

dramatically increases the likelihood of finding a high-quality fit resource faster.

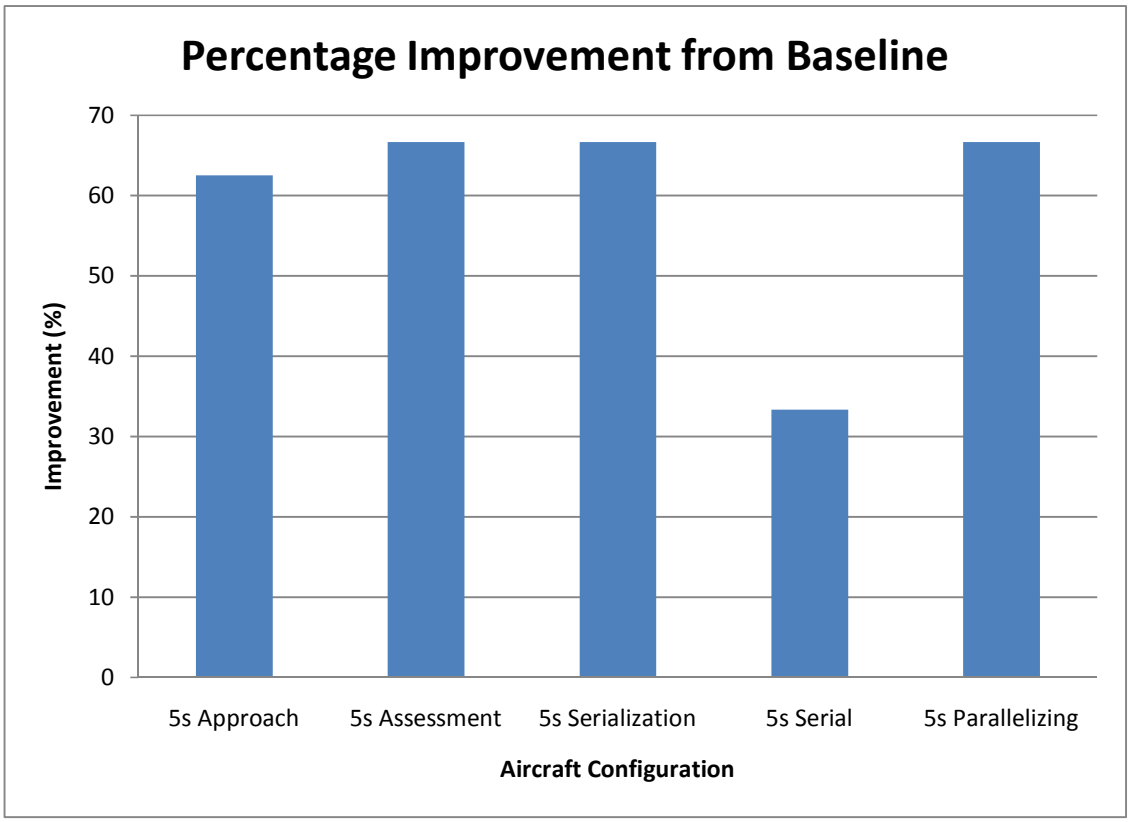

Figure 43 - Percent improvement using DARTS

Even the addition of failed caches (i.e., one or two out of four nodes) did not prevent the system from accomplishing its reallocation goals; in some instances, performance was reduced, but even the worst-recorded value still represented a 17% improvement over baseline. Figure 46 shows percent improvement under the different conditions (i.e., fully functional, single-cache and dual-cache failures) together. In the 5s serial configuration, the dual-cache failure actually scores higher because the nodes experiencing caching failures were randomly selected, even between trials; therefore, the position and selection of the node, in this case, has a greater effect on overall performance that the strict number of nodes with inoperative caches.

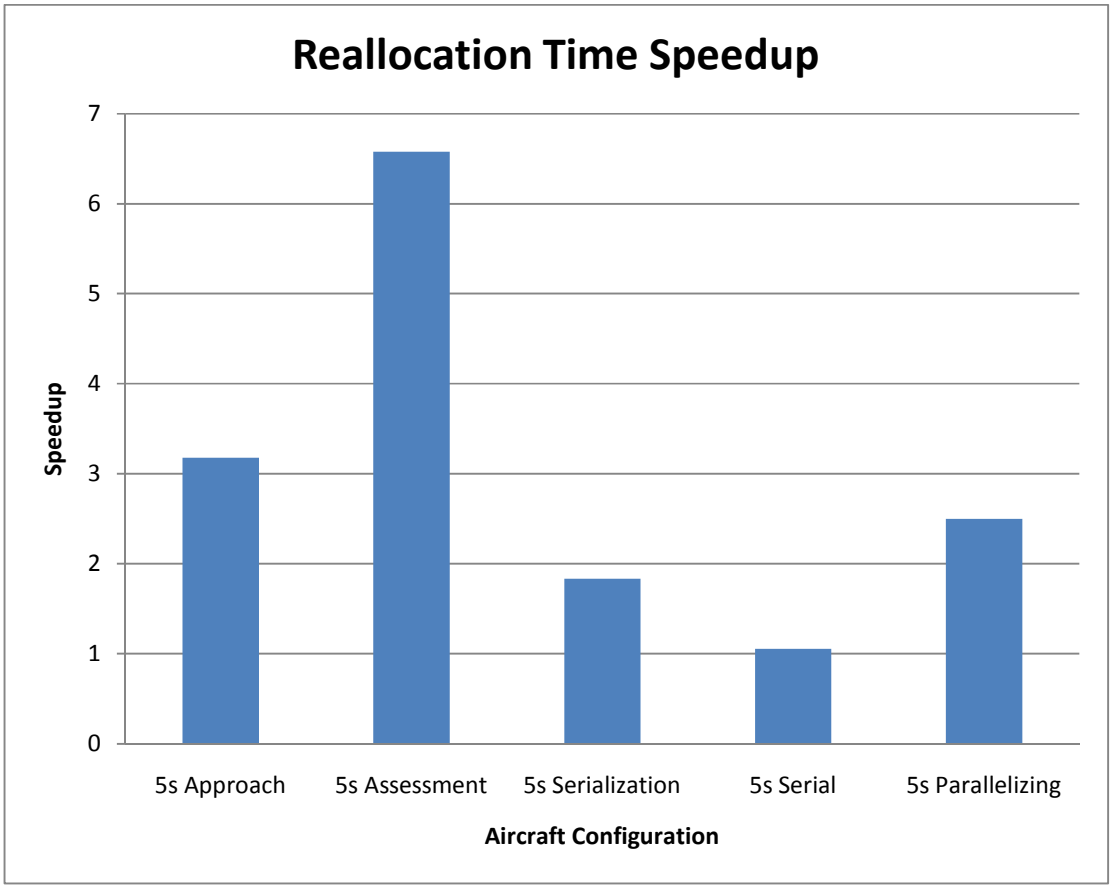

Figure 44 - Speedup using DARTS

Speedup also followed the predictable patterns of the number of messages required; in the 5s assessment case, speedup was over 6, fitting with the conclusion reached in Chapter V; the network was more densely connected (as opposed to serial), and therefore experiences a higher improvement. Likewise, the least-benefitting configuration in terms of speedup (1.06) was the serial link communication, also fitting the conclusions reached in Chapter V. As a result, in every functional case, the configuration that benefitted the least from the caching algorithm was, predictably, the serial case. Here, a caching algorithm can save one or two message propagations, as serial links forward end-to-end. Worst-case, the cache only blocks one

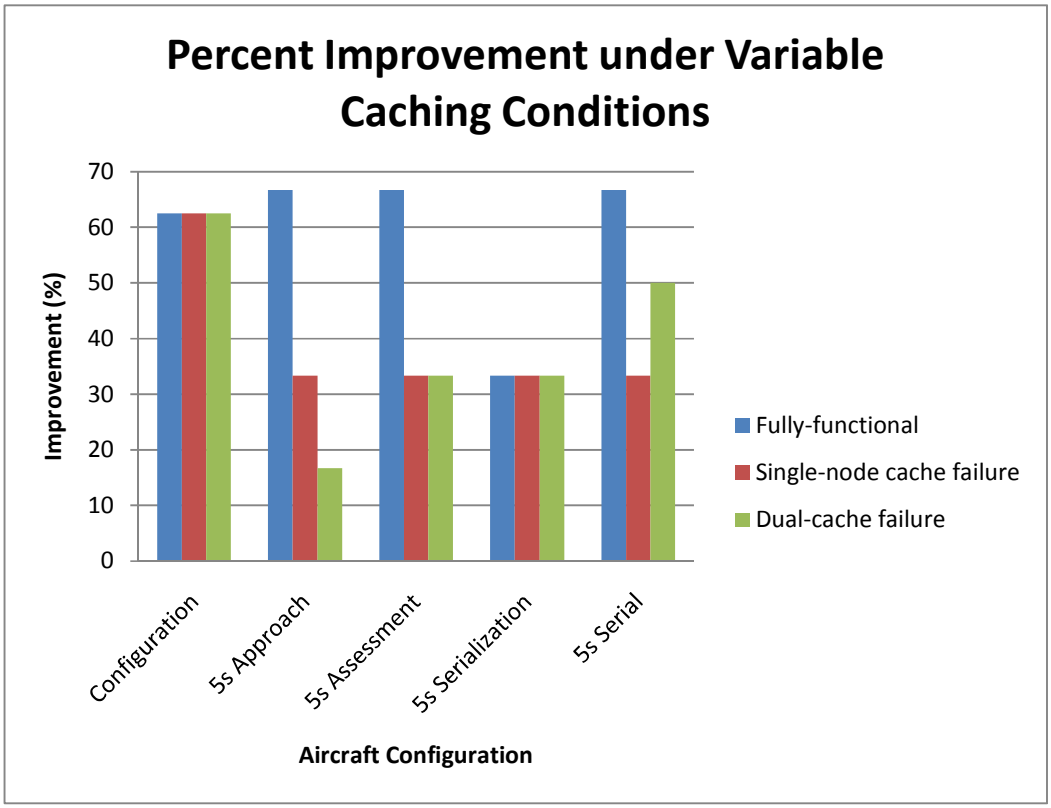

Figure 45 - Percent improvement using DARTS with 0, 1, and 2-node cache failures

propagation at the very extreme of the serial chain. This behavior was expected prior to simulation, and the data confirm this suspicion. Still, because DARTBOARD

encompasses a relatively small number of aircraft compared with large MANETs, a one-link propagation block can still register as a significant improvement, all things considered.

When looking at the reallocation speedup and message reduction in DARTBOARD, we must remind ourselves what this means in context. If the local connections between aircraft were gigabit Ethernet connections and the aircraft's onboard computers eight-core CISC-based CPUs, the reallocation task would be a feat so miniscule and menial that the entire argument of this dissertation would be invalidated. However, because the onboard computers on each aircraft use low-power 32-bit RISC CPUs, a meager amount of memory, and local links with 250 kilobit/second maximum data speeds, assisting the reallocation effort through intelligent caching algorithms and discreet message footprints becomes more important and useful. For the 5s assessment pass configuration, for example, the total local bandwidth becomes 7 links * 250 kilobits/second maximum. However, in practice, the radio throughput is divided among the number of connections per node, giving us an overall theoretical bandwidth of 5 nodes * 250 kb/s = 1.25 Mbit/s maximum. When multiplied by a "realism" factor of 80% (chosen completely arbitrarily) we can expect an average of 1 Mbit/second overall throughput. When using the DARTS method of reallocation, we see that the maximum time to reallocate under the baseline condition was 0.69 seconds under 5s approach configuration case. With 8 messages required at 256 bits each, this gives us a reallocation overhead (not including any other data running on the network) of 8 * 256 bits = 2048 bits (256 bytes) of data during the reallocation process. Instead, for the cached approach, we use 3 * 256 bits = 768 bits (96 bytes) during the reallocation process. When

considering the global bandwidth loss mentioned in Chapter III, this metric is affected not by node failure, but rather by reallocation overhead. In this case, the global bandwidth less is insignificant – less than 0.1%. In the cached case, the process took only 0.21 seconds, meaning that the reallocation speed and "bandwidth" is significantly higher. Small numbers either way, but supposing a larger MANET, of say 64 nodes, the savings become immediately apparent; while the size of the MANET varies, the speed of the links cannot, making the impact of the method even greater. In addition, the significant speedup (over 6 in some cases) means that the reallocation method not only uses less data (in terms of the number of messages) but also requires significantly less time, reducing the average impact on the network even further by making its process a brief event.

CHAPTER IX

CONCLUSIONS AND FUTURE WORK

Combined with a triggering mechanism, like the HybrIDS implementation featured here, DARTS provides a powerful preemptive problem solving architecture that forms the core of the DARTBOARD fault-tolerant disaster area assessment platform targeted at F2 tornados and above in its design specifications. Quite simply, no other distributed and preemptive fault-tolerant mechanism exists as of the time of this writing. With life-saving capabilities, the promise of reduced costs, the elimination of a single point of failure, and a faster, wider, heterogeneous and more detailed way to scan for survivors and structural damage, this author believes that DARTBOARD succeeds in providing benefits like no other system. With future work and development concentrated on bringing the multiple-aircraft scenario out of the world of simulation and into field tests, the potential of this system to develop into other applications is apparent.

This dissertation research sought to develop a resource reallocation mechanism to provide fault-tolerance to homogeneous and semi-homogeneous networks. This was effectively achieved through the implementation of DARTS, which also made such a reallocation system more efficient and capable of understanding application-level resource requirements. Through testing and optimization, the work has shown that the greatest gains to be made stand in densely-populated networks, where a larger degree of interconnectivity exists. Lastly, the successful integration of DARTS into a viable application along with a triggering mechanism was accomplished through the integration of DARTS into a distributed sensing application involving multiple airborne disaster area

assessment aircraft. The author believes that the system demonstrates potential for use in a number of distributed sensing applications where resource redundancy is present. Because of the significant time reduction (up to 6 times speedup) and minimization of the reallocation flood (a reduction of up to 75% of messages needed) was achieved, a sound case and proof of concept has been demonstrated and highlights the system's adaptability and potential.

The investigation into clustering density demonstrated that this metric does in fact alter how effectively DARTS, the reallocation solution in this dissertation, can operate. The test results of both the generic DARTS testing and application-specific DARTBOARD efficiency tests show that serial, sparse network configuration stand, on average, to benefit the least from the cached reallocation mechanism.

As mentioned earlier, work must be done to integrate multiple airframes together to perform in-field testing of the triggering mechanism, reallocation methods, and the ability for aircraft to stay cohesive during their assessment runs. The cooperation and launching of multiple aircraft into a pattern alone involves considerable planning and work to accomplish. However, because the Paparazzi platform offers flexibility of implementation due to its open-source and modular design, coordination is a feasible concept and should be explored.

Furthermore, once a functioning multi-aircraft platform is operational, the level of testing must far exceed the number of trials run in simulation for this dissertation. An appropriate boundary exploration study is necessary, and will be performed in the future. Also, because error conditions of the IDS (outside of the scope of this work) can affect

the operation and efficiency of DARTBOARD (i.e., reallocation prior to a failure is preferable to reallocation after a failure) the efficacy of HybrIDS in this context must be identified in more detail, with possible functional limitations exposed in order to better tune its operation and ensure deterministic error handling and improved efficiency. For the dissertation work, it performed perfectly; no errors were identified during its operation, as its parameters were strictly specified. However, it is likely that in the field, conditions that were not accounted for during simulation may exist, requiring further modification of the triggering mechanism.

Regardless of the direction this research will take, it holds the promise of changing how post-storm disaster area assessment is done. What the future holds for DARTBOARD, especially if included in collaborative work with the National Oceanic and Atmospheric Administration, the National Severe Storms Lab, the National Storms Prediction Center, and collective efforts such as the Verification of the Origins of Rotation in Tornadoes Experiment (VORTEX), is unknown, but likely promising. This author believes that DARTBOARD hits a bull's eye when considering a capable, all-inclusive ready-to-fly solution that is ready to take on the future challenges it may face.

APPENDIX A

AVIONICS AND GROUND CONTROL

This appendix is designed to aid the reader in understanding the components onboard the Alpha 40-implementation of DARBOARD. Described are the Paparazzi Autopilot system, how it works, configuration settings, and interface to the Ground Control Station (GCS).

Avionics

The selection of an exceptional candidate airframe provides aircraft stability, extended runtimes, and the flexibility to operate in variable environments. Each aircraft/node in the DARTBOARD system configuration must be capable of autonomous flight in order to proceed through an assessment procedure (multiple aircraft would require multiple pilots, yielding coordination, expense, and accuracy problems). To this end, two candidate autopilot systems and navigation computers were evaluated based on performance, flexibility of future customized modules and control methods, and the comprehensive nature of the air and ground control systems. The first, called the Paparazzi Tiny 2.11 [92], is an open-source (hardware and software) platform with primary research originating at ENAC in France. Its flight computer is based on a 32-bit ARM7 microcontroller, featuring 8 kilobytes of RAM. The second autopilot, called the ArduPilot [93], is based on the Arduino micro development board, a popular microcontroller platform among hobbyists and embedded systems enthusiasts (including

the author). It runs on an Atmega 168 16-bit microcontroller with 4 kilobytes of RAM and 16 kilobytes of EEPROM.

Both the ArduPilot and the Paparazzi project provided end-to-end hardware and software support (though "support" must be used loosely, as considerable time was spent compiling, creating, editing, and tweaking configuration files for the airframe, radio system, and flight paths), and accept a variety of input sensors (including GPS) in order for the host aircraft to fly efficiently and within its target zones. Both systems incorporate inner and outer proportional–integral–derivative controller (PID) control loops for actuating the host aircraft's control surfaces. Also, both systems feature a variety of input capabilities, such as the use of infrared thermopiles, pitot tubes, MEMS-based accelerometer gyroscopes, and GPS receivers. Both also can implement bidirectional wireless control to a ground control station (GCS) by means of medium to long-range 802.15.4 radio modems, interfaced by a serial or direct protocol. After careful consideration, the Paparazzi project provided a more powerful computing platform, coupled with a well-developed and tested control architecture. However, it required significantly more development and implementation time – taking several weeks to fully prepare an airframe for flight.

Once the Alpha's airframe parameters were input (control methods, radio communication modes, servo actuation throws and limits, directions, and number and configuration of control surfaces), the airframe was calibrated in-flight to allow optimum efficiency with ideal course tracking. PID control gains were adjusted rigorously to minimize oscillations. The Paparazzi aircraft configuration calls for the use of a ground

control station (GCS) in addition to the aircraft's 2.4 GHz receiver link. In this configuration, the R/C radio receiver in the airplane outputs to a PPM encoder board that then relays the received control inputs to the autopilot (AP) computer board. The autopilot, when operating in "manual" mode, relays the signals directly to the servomotors, to actuate the control surfaces. In this mode of operation, Paparazzi acts transparently, only as a mediator between the receiver and the servos.

A three-way toggle on the R/C radio (or a command from the GCS) is used to activate the AP's autonomous flight modes. Mode 1 is an "enhanced stability mode", incorporating some feedback from the control sticks on the R/C radio (a Spektrum DX7 7-channel 2.4 GHz DSSS DSM2 digital computer radio) but maintaining constant altitude and stable flight characteristics. Mode 2, used most often, initiates autonomous waypoint navigation, as specified in waypoint files that are preloaded and/or updated on-the-fly by the GCS. The GCS itself utilizes the AP board's serial link by means of a 900 MHz 802.15.4 radio (XBee MaxStream) pair. For its role in the distributed sensing application, we will refer to this link as the *Ground Link*. In Mode 2, no control stick inputs on the DX7 cause an effect on the control surfaces, unless this is specified ahead of time. For instance, the Alpha can be steered using only ailerons (and applying corrective elevator to pull the aircraft around the turn), by using only the rudder (skidding or sliding the aircraft around by the tail), or by using coordinated turns (employing both the ailerons to bank and the rudder to pull around the turn). While coordinated is the most stable flight mode, the AP has been programmed to use aileron steering. This way, the rudder can be used to override for custom course corrections. In practice, this has become completely unnecessary, as the three-way toggle can disable the autopilot. In addition, flight

characteristics in AP Mode 2 are so stable, that even on gusty, windy days, the aircraft flies to its intended coordinates.

Attitude sensing was accomplished by integrating a set of horizontal and vertical thermopiles. Four orthogonally-oriented thermopiles are pointed normal to the vertical axis. By measuring far-infra-red light, these devices can understand the pitch of the plane by performing a differential temperature assessment. In theory, and in practice, the ground is always warmer than the sky. Even on cloudy and snowy days, this holds true. If the aircraft is steeply banked, there will be a significant temperature differential between the left-side and right-side set of thermopiles. The amount of this difference corresponds to the bank angle. The same holds true for pitch control. The vertical thermopiles are used to assess absolute orientation (whether or not the plane flight right-side-up or inverted) and to determine to some degree the aircraft's relative altitude (a weaker ground temperature indicates higher altitude, when measured continuously from the start of the flight (assuming it was launched on the ground.) Figure 34 shows this configuration. The Paparazzi project includes modules for the integration of accelerometers for absolute positioning and fast bank angle detection. This has proven unnecessary because the Alpha is a large plane, and therefore moves gracefully and deliberately, with less oscillation in its movements. Also, experimentally, flight path following was accurate and needed no further assistance in the form of gyros.

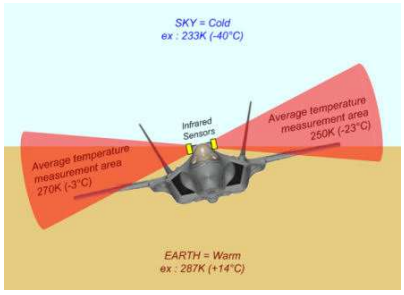

Figure 46 - Thermopile orientation

Flight-path monitoring, on the other hand, relies exclusively on GPS for altitude, and absolute position information. While the thermopiles can detect the attitude and angle of attack of the plane, this alone provides little or no information on the aircraft's position in a 3-dimensional geodetic coordinate system. The Paparazzi Tiny 2.11 AP board utilizes a Ublox LEA-5H differential GPS that can provide 2 Hz update (most GPS units only provide 1 Hz updates, a general limitation imposed by the time scale used in global positioning system satellites. The LEA-5H overcomes this limitation by using differential integration.) Once an airframe and radio configuration XML files have been established, the user must define a flight plan configuration file.

The flight plan file is organized into two sections. The first lists coordinates of waypoints, while the second is comprised of "blocks" that lists the order and routing method for those waypoints. For instance, there may exist three co-linear waypoints, but the routing blocks may direct the aircraft to fly them non-sequentially. Both the waypoint and routing block sections can specify altitude and operational methods. "For" loops can be integrated into the routing blocks, as well as conditional data, for advanced navigation patterns. Furthermore, the routing blocks can be updated on-the-fly through the GCS.

Ground Control

The GCS provides a tactical overview of flight path, error, (the difference between the intended path and the actual path) flight statistics (such as speed, altitude, heading, climb rate, and attitude) and control over the flight plan, among many other duties. When properly implemented, the GCS is capable of fully controlling a UAV utilizing the Paparazzi system, if such intervention is desired. Its primary use during

setup and configuration is for tweaking airframe and configuration parameters during test flights; because the GCS is connected to the UAV's AP by the use of a 900 MHz XBee radio, such updates are possible. However, because final updates to the flight computer must be made in the form of compiled, uploadable firmware files that must be flashed to the microcontroller's EEPROM, these changes are only stored in volatile memory until the system is powered off. The GCS also displays the waypoints and the UAV's present position in real time, overlaying the map with tiles from the Google Maps Geographic Information System (GIS). Figure 35 shows the GCS during a simulated run.

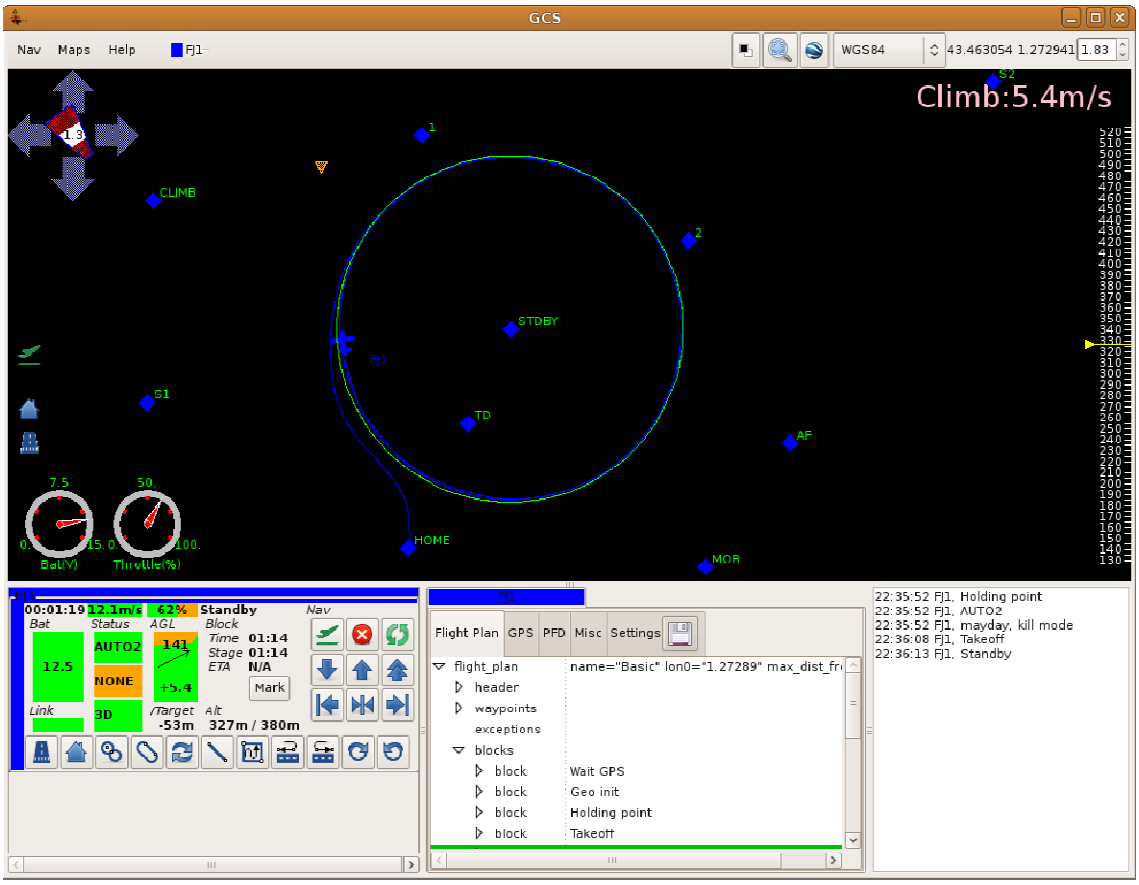

Figure 47 - The Paparazzi Ground Control Station (CGS)

Software Simulation

The GCS is also used to execute a simulation run; because configuring an airframe and testing a flight plan is risky on a sophisticated airframe, the Paparazzi airframe permits two types of simulation: Software-In-The-Loop (SITL) and Hardware-In-The-Loop (HITL). SITL implements all the airframe configuration files and flight plans, and compiles this into an x86-executable simulation on the GCS computer (running an Ubuntu or Debian Linux distribution). Execution is done identically to the AP board's firmware. The second option utilizes the AP board's CPU to execute and return control decisions. In this case, the GCS merely displays the outputs of the AP board's hardware. This simulation method was not used; airframes and flight plans were preliminarily tested in SITL simulation and then implemented on the actual airframe.

APPENDIX B

FLIGHT CONDITION LIMITATIONS

This appendix lists information that can affect flight performance of the DARTBOARD implementation. Of particular issue are wind speeds immediately following a storm. Listed here are maximum operational wind speeds and associated characteristics that can alter whether or not the system is capable of canvassing the disaster are.

Flight Condition Limitations

A potential obstacle to navigation and general operation of the aircraft is wind speed, and in particular, unpredictable gusts. Because the Alpha has a large wing camber and oversized wing area, it is capable of lift at speeds as low as 6 meters per second, or 24 kilometers per hour; in gliding conditions, this speed can drop even lower. Because wind speed must be used to choose the takeoff direction (to maximize lift, and therefore flow-rate of air over the wing surface, planes should take off facing into the wind), runways or open fields may not provide sufficient takeoff length in ideal conditions. Though downwind takeoffs are possible, strong winds could prevent lift and cause unpredictable takeoffs and stalls, or worse, longer takeoff length causing the plane to strike an obstacle before achieving lift. This particular problem can be solved by hand-launching the aircraft. This technique is popular with smaller aircraft, but may be difficult to accomplish with the Alpha, and especially the Telemaster, due to weight concerns and possible force exerted by the human arm. Still, in un-ideal conditions, this offers a feasible compromise.

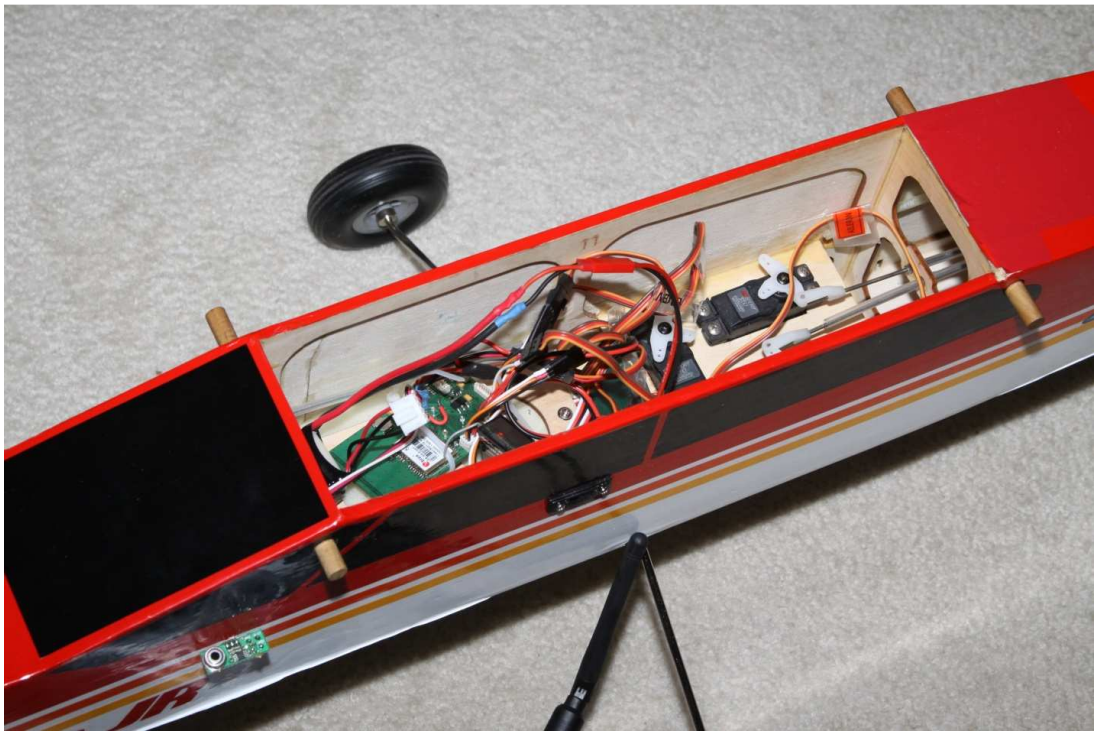

Figure 48- The first Alpha with Autopilot and vertical thermopile and servos

Barring a failed takeoff, wind can prevent the aircraft from reaching its destination; the Paparazzi AP is capable of performing real-time throttle management. The author has had some experience with overpowering winds; a prior version of the Alpha 40 UAV suffered catastrophic damage as wind proved to be too strong for the engine to pull the plane forward; the over-sized wing caused the plane to rotate, pointing the nose straight toward the ground, just a meter or two off the ground. While the fuselage was not salvageable, careful design and positioning of the AP's electronics allowed them to be salvaged and re-used in the newer, electric version of the Alpha. Figure 37 shows the positioning of the autopilot board (on the old, no-longer existing Alpha) and associated control electronics. The Alpha can safely fly in winds not exceeding 28 km/h. Stronger winds, or the presence of gusts can put the entire flight (and bystanders) at risk. Because the propeller mounted on the front of the airframe is powered

by an 800 W motor at speeds of up to 9000 RPM, it is capable of causing severe injury or death if it contacts a person while spinning. Still, the danger is less than that of a rotary-wing aircraft, such as a helicopter, which requires much larger rotating surfaces, and is inherently less stable than an airplane. A useful bit of information, for the R/C pilot, for the survey, and for bystanders, is that according to [94], winds can variable or calm within minutes of the passage of a thunderstorm – tornadic or otherwise. Gust fronts ahead of the storm and other storm-related phenomena precede the weather front or supercell, leaving calmer air in its wake, as most of the atmospheric energy has been dissipated by this point. However it is still possible for secondary storm lines or cells to follow behind the first. Winds in this case would be dominated by the secondary storm's gust front, posing a potential problem to UAV-based flight.

REFERENCES

1. J. Langenwalter, “Embedded automotive system development process - steer-by-wire system,” *Design, Automation and Test in Europe*, pp. 538-539 Vol. 531.

2. E.A. Lee, “Absolutely positively on time: what would it take?,” *Computer*, vol. 38, no. 7, 2005, pp. 85-87.

3. K. Walther, and J. Nolte, “A Flexible Scheduling Framework for Deeply Embedded Systems,” *21st International Conference on Advanced Information Networking and Applications Workshops, AINAW '07.* , pp. 784-791.

4. M. Barr, “Real men program in C,” 2009; http://embedded.com/design/218600142.

5. A. Garcia-Martinez, J.F. Conde, and A. Vina, “A comprehensive approach in performance evaluation for modern real-time operating systems,” *22nd EUROMICRO Conference EUROMICRO 96.*, pp. 61-68.

6. G. Karsai, C. Biegl, S. Padalkar, J. Sztipanovits, K. Kawamura, N. Miyasaka, and M. Inui, “Knowledge-Based Approach to Real-Time Supervisory Control,” *American Control Conference, 1988*, pp. 620-625.

7. Z. Yang, L. Jie, and E.A. Lee, “A Programming Model for Time-Synchronized Distributed Real-Time Systems,” *13th IEEE Real Time and Embedded Technology and Applications Symposium, RTAS '07.* , pp. 259-268.

8. Z. Gao, Y. Yang, J. Zhao, J. Cui, and X. Li, “Service Discovery Protocols for MANETs: A Survey,” *Mobile Ad-hoc and Sensor Networks*, Lecture Notes in Computer Science 4325/2006, Springer Berlin / Heidelberg, 2006, pp. 232-243.

9. A.N. Mian, R. Baldoni, and R. Beraldi, “A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks,” *Pervasive Computing, IEEE*, vol. 8, no. 1, 2009, pp. 66-74.

10. C. Jo Woon, H. Ho Young, J. Chang Yong, and S. Dan Keun, “Analysis of Throughput and Energy Consumption in a ZigBee Network Under the Presence of Bluetooth Interference,” *IEEE Conference on Global Telecommunications Conference, GLOBECOM '07.*, pp. 4749-4753.

11. A. Rajeswaran, K. Gyouhwan, R. Negi, and N. Sai Shankar, “Interference Handling in UWB Versus 802.11n Networks,” *IEEE International Conference on Communications, ICC '07* pp. 4710-4715.

12. Z. Alliance, “ZigBee Home Automation: The New Global Standard for Home Automation,” *Book ZigBee Home Automation: The New Global Standard for Home*

Automation, Series ZigBee Home Automation: The New Global Standard for Home Automation, 2007, pp.

13. D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, "GSD: A Novel Group-based Service Discovery Protocol for MANETS," *Book GSD: A Novel Group-based Service Discovery Protocol for MANETS*, Series GSD: A Novel Group-based Service Discovery Protocol for MANETS, ed., Editor ed.^eds., University of Maryland, Baltimore County, 2000, pp.

14. S. Jian, and G. Wei, "A survey of service discovery protocols for mobile ad hoc networks," *International Conference on Communications, Circuits and Systems, ICCCAS 2008*, pp. 398-404.

15. R. Zoican, "Analysis of Scalability in MANET's Protocols," *8th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, TELSIKS 2007*, pp. 605-608.

16. T.K. Paul, and T. Ogunfunmi, "Wireless LAN Comes of Age: Understanding the IEEE 802.11n Amendment," *IEEE Circuits and Systems Magazine*, vol. 8, no. 1, 2008, pp. 28-54.

17. J. Lorincz, and D. Begusic, "Physical layer analysis of emerging IEEE 802.11n WLAN standard," *8th International Conference Advanced Communication Technology, ICACT 2006.* , pp. 6 pp.-194.

18. T. Selvam, and S. Srikanth, "Performance study of IEEE 802.11n WLANs," *First International Communication Systems and Networks and Workshops, COMSNETS 2009*, pp. 1-6.

19. D.D. Perkins, H.D. Hughes, and C.B. Owen, "Factors affecting the performance of ad hoc networks," *IEEE International Conference on Communications, ICC 2002.* , pp. 2048-2052 vol.2044.

20. H.A. Amri, M. Abolhasan, and T. Wysocki, "Scalability of MANET routing protocols for heterogeneous and homogenous networks," *Computers & Electrical Engineering*, vol. In Press, Corrected Proof.

21. C.E. Perkins, and E.M. Royer, "Ad-hoc On-Demand Distance Vector Routing," *Book Ad-hoc On-Demand Distance Vector Routing*, Series Ad-hoc On-Demand Distance Vector Routing, Sun Microsystems Laboratories

University of California, Santa Barbara, 1997, pp.

22. H. Zhou, "A Survey on Routing Protocols in MANETs," *Book A Survey on Routing Protocols in MANETs*, Series A Survey on Routing Protocols in MANETs, ed., Editor ed.^eds., Michigan State University, 2003, pp.

23. A. Patwardhan, J. Parker, M. Iorga, A. Joshi, T. Karygiannis, and Y. Yesha, "Threshold-based intrusion detection in ad hoc networks and secure AODV," *Ad Hoc Networks*, vol. 6, no. 4, 2008, pp. 578-599.

24. J.S. Khoury, H. Jerez, and L.D. Cicco, "Design and implementation of a framework for persistent identification and communication in emerging networks," *Book Design and implementation of a framework for persistent identification and communication in emerging networks*, Series Design and implementation of a framework for persistent identification and communication in emerging networks, ed., Editor ed.^eds., ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp.

25. A. Boukerche, and S.K. Das, "Congestion control performance of R-DSDV protocol in multihop wireless ad hoc networks," *Wirel. Netw.*, vol. 9, no. 3, 2003, pp. 261-270.

26. J.B.D. Cabrera, C. Gutierrez, and R.K. Mehra, "Infrastructures and algorithms for distributed anomaly-based intrusion detection in mobile ad-hoc networks," *IEEE Conference on Military Communications, MILCOM 2005.*, pp. 1831-1837.

27. L. Keum-Chang, and L. Mikhailov, "Intelligent intrusion detection system," *2nd International IEEE Conference Intelligent Systems*, pp. 497-502 Vol.492.

28. A.P. Lauf, "HybrIDS: Embeddable Intrusion Detection System," Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN, 2007.

29. A.P. Lauf, R.A. Peters, and W.H. Robinson, "A distributed intrusion detection system for resource-constrained devices in ad-hoc networks," *Ad Hoc Networks*, vol. In Press, Corrected Proof, 2009.

30. N. Marchang, and R. Datta, "Collaborative techniques for intrusion detection in mobile ad-hoc networks," *Ad Hoc Networks*, vol. In Press, Corrected Proof.

31. A. Mishra, K. Nadkarni, and A. Patcha, "Intrusion detection in wireless ad hoc networks," *IEEE Wireless Communications*, vol. 11, no. 1, 2004, pp. 48-60.

32. K. Nadkarni, and A. Mishra, "Intrusion detection in MANETS - the second wall of defense," *29th Annual Conference of the IEEE Industrial Electronics Society, IECON '03*, pp. 1235-1238 Vol.1232.

33. A. Patwardhan, J. Parker, M. Iorga, A. Joshi, T. Karygiannis, and Y. Yesha, "Threshold-based intrusion detection in ad hoc networks and secure AODV," *Ad Hoc Networks*, vol. In Press, Corrected Proof.

34. R. Puttini, J.M. Percher, L. Me, and R. de Sousa, "A fully distributed IDS for MANET," *Ninth International Symposium on Computers and Communications, ISCC 2004* pp. 331-338 Vol.331.

35. S.A. Razak, S.M. Furnell, N.L. Clarke, and P.J. Brooke, "Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks," *Ad Hoc Networks*, vol. In Press, Corrected Proof.

36. S.A. Razak, S.M. Furnell, N.L. Clarke, and P.J. Brooke, "Friend-assisted intrusion detection and response mechanisms for mobile ad hoc networks," *Ad Hoc Networks*, vol. 6, no. 7, 2008, pp. 1151-1167.

37. J. Voelcker, "Top 10 tech cars [hybrid electric vehicles]," *IEEE Spectrum*, vol. 42, no. 3, 2005, pp. 22-30.

38. D. Watkins, and C. Scott, "Methodology for evaluating the effectiveness of intrusion detection in tactical mobile ad-hoc networks," *IEEE Conference on Wireless Communications and Networking Conference, WCNC 2004*, pp. 622-627.

39. Z. Xiaodong, H. Zhiqiu, and Z. Hang, "Design of a Multi-agent Based Intelligent Intrusion Detection System," *1st International Symposium on Pervasive Computing and Applications*, pp. 290-295.

40. A. Yamada, Y. Miyake, K. Takemori, and T. Tanaka, "Intrusion detection system to detect variant attacks using learning algorithms with automatic generation of training data," *International Conference on Information Technology: Coding and Computing, ITCC 2005.*, pp. 650-655 Vol. 651.

41. Y. Zhenwei, J.J.P. Tsai, and T. Weigert, "An Automatically Tuning Intrusion Detection System," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 37, no. 2, 2007, pp. 373-384.

42. D. Allard, and J. Hutcherson, "Communications Across Complex Space Networks," *Aerospace Conference, 2008 IEEE*, pp. 1-11.

43. C. Daskalakis, and P. Martone, "Alternative surveillance technology for the Gulf of Mexico," *The 23rd Digital Avionics Systems Conference,DASC 04.* , pp. 1.D.4-1.1-8 Vol.1.

44. W.H. Harman, "ADS-B airborne measurements in Frankfurt," *The 21st Digital Avionics Systems Conference*, pp. 3A3-1-3A3-11 vol.11.

45. D.S. Hicok, and D. Lee, "Application of ADS-B for airport surface surveillance," *The 17th AIAA/IEEE/SAE Digital Avionics Systems Conference*, pp. F34/31-F34/38 vol.32.

46. R. Nichols, D.J. Bernays, T. Spriesterbach, and V. Dongen, “Testing of traffic information service broadcast (TIS-B) and ADS-B at Memphis International Airport,” *The 21st Digital Avionics Systems Conference*, pp. 3A2-1-3A2-13 vol.11.

47. Y. Pizhou, and L. Chaodong, “A RISC CPU IP core,” *2nd International Conference on Anti-counterfeiting, Security and Identification, ASID 2008.* , pp. 356-359.

48. K. Sung Ho, L. Seung Ho, C. Byeong Yoon, and L. Moon Key, “A 32-bit low power RISC core for embedded applications,” *IEEE Region 10 International Conference on Microelectronics and VLSI, TENCON '95.*, , pp. 488-491.

49. G. Thomas, “Benefits of performance standards for lithium-ion and lithium-ion polymer batteries,” *The Seventeenth Annual Battery Conference on Applications and Advances*, pp. 223-225.

50. D. Ganapathy, and E.J. Warner, “Defining thermal design power based on real-world usage models,” *11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems. ITHERM 2008.* , pp. 1242-1246.

51. A.H. Han, H. Young-Si, A. Young-Ho, L. So-Jin, and C. Ki-Seok, “Virtual ARM Platform for embedded system developers,” *International Conference on Audio, Language and Image Processing, ICALIP 2008.*, pp. 586-592.

52. Texas Instruments, “OMAP 3 Processors: OMAP3440,” http://focus.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&navigationId=12796&contentId=36505.

53. G. Dhillon, *Principles of Information Systems Security: Texts and Cases*, Wiley, 2006.

54. A.P. Lauf, and W.H. Robinson, “Fault Tolerance in MANETs Using a Task-to-Resource Reallocation Framework,” *International Conference on Computational Science and Engineering, CSE '09.* , pp. 753-758.

55. L. Xiao, C. Xu, and W. Zhong, “Performance Analysis of the DSSS System Based on Sequence Pairs,” *First International Conference on Communications and Networking in China, ChinaCom '06.* , pp. 1-4.

56. Y. Kazuaki, C. Wenxi, and W. Daming, “An Intensive Survey of 3G Mobile Phone Technologies and Applications in Japan,” *Sixth IEEE International Conference on Computer and Information Technology, CIT '06.* , pp. 265-265.

57. E. Chikuni, and M. Dondo, “Investigating the security of electrical power systems SCADA,” *AFRICON 2007*, pp. 1-7.

58. P. Kocher, R. Lee, G. McGraw, A.A.R.A. Raghunathan, and S.A.R.S. Ravi, "Security as a new dimension in embedded system design," *41st Design Automation Conference*, pp. 753-760.

59. J. Li, L. Da-You, and Y. Bo, "Smart home research," *International Conference on Machine Learning and Cybernetics*, pp. 659-663 vol.652.

60. D.R. Choudhary, D. Anshul, S. Roy, and C.S. Thejaswi, "Computationally and Resource Efficient Group Key Agreement for Ad Hoc Sensor Networks," *2nd International Conference on Communication Systems Software and Middleware, COMSWARE 2007.*, pp. 1-10.

61. C. Day, "Quantum Computing Is Exciting and Important--Really!," *Computing in Science & Engineering*, vol. 9, no. 2, 2007, pp. 104-104.

62. J. Henkel, and L. Yanbing, "Avalanche: an environment for design space exploration and optimization of low-power embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 4, 2002, pp. 454-468.

63. A. Patcha, and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, 2007, pp. 3448-3470.

64. G.E. Reeves, and J.F. Snyder, "An overview of the Mars exploration rovers' flight software," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1-7 Vol. 1.

65. Y.-A. Huang, and W. Lee, "Attack Analysis and Detection for Ad Hoc Routing Protocols," *Recent Advances in Intrusion Detection*, Lecture Notes in Computer Science 3224/2004, Springer, 2004, pp. 125-145.

66. O. Kachirski, and R. Guha, "Effective intrusion detection using multiple sensors in wireless ad hoc networks," *36th Annual Hawaii International Conference on System Sciences*, pp. 8 pp.

67. A.P. Lauf, Peters, R. A. and Robinson, W. H., "Intelligent Intrusion Detection: A Behavior-Based Approach," *The 21st Advanced Information Networking and Applications: Symposium for Embedded Computing, 2007.*

68. E. Naess, D.A. Frincke, A.D. McKinnon, and D.E. Bakken, "Configurable middleware-level intrusion detection for embedded systems," *25th IEEE International Conference on Distributed Computing Systems Workshops*, pp. 144-151.

69. R.L. Rivest, A. Shamir, and L.M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Massachusetts Institute of Technology, Laboratory for Computer Science, 1977.

70. M. Borsc, and H. Shinde, "Wireless security & privacy," *IEEE International Conference on Personal Wireless Communications, ICPWC 2005.*, pp. 424-428.

71. L. Chih-Chung, and T. Shau-Yin, "Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter," *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 277-285.

72. Q. Yang, X. Wu, R. Zhou, and R. Lu, "An embedded RSA processor for encryption and decryption," *4th International Conference on ASIC*, pp. 356-359.

73. H. Seung-Jo, O. Heang-Soo, and P. Jongan, "The improved data encryption standard (DES) algorithm," *IEEE 4th International Symposium on Spread Spectrum Techniques and Applications*, pp. 1310-1314 vol.1313.

74. Inno-Logic, "AES CCM Encryption and Decryption," *Book AES CCM Encryption and Decryption*, Series AES CCM Encryption and Decryption, ed., Editor ed.^eds., 2008, pp.

75. O.O. Khalifa, M.D.R. Islam, S. Khan, and M.S. Shebani, "Communications cryptography," *RF and Microwave Conference, RFM 2004*, pp. 220-223.

76. S. Buchegger, and J.Y. Le Boudec, "Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks," *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*.

77. S. Buchegger, and J.Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol," *3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing*.

78. S. Buchegger, and J.Y. Le Boudee, "Self-policing mobile ad hoc networks by reputation systems," *Communications Magazine, IEEE*, vol. 43, no. 7, 2005, pp. 101-107.

79. , "Swarming for Success," *Book Swarming for Success*, Series Swarming for Success, NASA Ames, 2005, pp.

80. P. Basu, K. Wang, and T.D.C. Little, "A novel approach for execution of distributed tasks on mobile ad hoc networks," *IEEE Wireless Communications and Networking Conference (WCNC2002)*, pp. 579-585.

81. W. Freeman, and E. Miller, “An experimental analysis of cryptographic overhead in performance-critical systems,” *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 348-357.

82. C. Yingying, W. Trappe, and R.P. Martin, “Detecting and Localizing Wireless Spoofing Attacks,” *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON '07.*, pp. 193-202.

83. L. Zhou, and Z.J. Haas, “Securing ad hoc networks,” *IEEE Network*, vol. 13, 1999, pp. 24-30.

84. C.E. Perkins, and E.M. Royer, “Ad-hoc on-demand distance vector routing,” *Second IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '99.* , pp. 90-100.

85. J. Kennedy, and R. Eberhart, “Particle swarm optimization,” *IEEE International Conference on Neural Networks*, pp. 1942-1948 vol.1944.

86. P.-Y. Yin, S.-S. Yu, P.-P. Wang, and Y.-T. Wang, “A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems,” *Computer Standards & Interfaces*, vol. 28, no. 4, 2006, pp. 441-450.

87. H.E. Brooks, “On the Relationship of Tornado Path Length and Wind to Intensity,” *Weather and Forecasting*, vol. 19, 2003, pp. 310-319.

88. SkyWarn, “SKYWARN Advanced Storm Spotter Guide,” http://spotterguides.us/advanced/advanced01.htm.

89. A.M. Society, *Glossary of Meteorology*.

90. T.T. Fujita, “Proposed characterization of tornadoes and hurricanes by area and intensity,” *Book Proposed characterization of tornadoes and hurricanes by area and intensity*, Series Proposed characterization of tornadoes and hurricanes by area and intensity, University of Chicago, 1971, pp.

91. , “Omnet++,” 4.0 ed., 2008

92. “The Paparazzi Autopilot Project,” http://paparazzi.enac.fr/.

93. “The ArduPilot Project.”

94. S. Vassiloff, National Severe Storms Laboratory