A Dissertation

entitled

Cyber Security Threat Analysis and Attack Simulation for Unmanned Aerial Vehicle Network

by

Ahmad Y. Javaid

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Engineering

Dr. Weiqing Sun, Committee Chair

Dr. Mansoor Alam, Committee Co-Chair

Dr. Vijay K. Devabhaktuni, Committee Member

Dr. Robert Green, Committee Member

Dr. Hong Wang, Committee Member

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo
August 2015

An Abstract of

Cyber Security Threat Analysis and Attack Simulation for
Unmanned Aerial Vehicle Network

by

Ahmad Y. Javaid

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the
Doctor of Philosophy Degree in Engineering

The University of Toledo
August 2015

Use of unmanned systems in various tasks has increased exponentially in the recent past. These systems enable users to complete vital missions efficiently, without risking human lives. Nonetheless, these systems pose a threat to the general population if the operational cyber security is not handled. Especially, the armed unmanned aerial vehicle systems (UAVS), which can cause catastrophic damage to life and property. It is important to know the risk and understand the impact of various possible attacks on the overall UAVS. Clearly, the most economical way to achieve this is to simulate operational scenarios of UAVS before actual deployment. In this dissertation, we propose methods to assess various threats, develop threat models, evaluate risk and impact of attacks. We finally use these methods to develop a simulation testbed environment for Unmanned Aerial Vehicle Networks (UAVNet) cyber security analysis. The testbed was designed to be open source to enhance the usability and audience reach. We also demonstrated the use of this testbed in academia for any related research or student learning and consequently, a performance evaluation of the testbed for use in generic computing environment was carried out. Based on the experiments performed for various communication denied scenarios, we evaluate the impact of various attacks against UAVNet from the communication perspective and report the results to demonstrate the necessity and usefulness of the simula-

tion testbed. Implemented attacks include Distributed Denial of Service (DDoS), Jamming, GPS Jamming and GPS Spoofing. Additional implementation of an anti-spoofing technique for GPS spoofing was further done to indicate the usefulness of testbed and flexibility to develop attacks as well as their detection and mitigation measures.

Dedicated to my parents

Mrs. Zubaida Khatoon and *Mr. Javaid Asrar*

— for their undying faith, trust and patience —

Acknowledgments

Although no amount of words can express my thankfulness and gratitude for my committee, my family and my friends, I would still make an attempt to acknowledge them.

I would like to express my deepest gratitude to my advisor, Dr. Weiqing Sun, for his guidance, care and patience. He not only provided me with an excellent environment for doing research, he also helped me mold myself into a better person by continuously supporting me at times when I was extremely lazy or unproductive. Dr. Mansoor Alam and Dr. Vijay Devabhaktuni have helped me not only understand the way of academic research but trained me to handle various challenges. I am very thankful to Dr. Robert Green and Dr. Hong Wang for guiding my research with critical feedback.

I would also like to thank Dr. Patricia Komuniecki, Dr. Susan Pocotte, Debbie, Tammy, Terri, Mary, Denise, Elissa, Teri, Brenda and the rest of the COGS for their constant support. Special thanks to my dear friends and colleagues, Farha and Quamar, for the constant brainstorming, suggestions, and feedback.

My parents have made me everything that I am today and my *American* parents, Cheryl and Eric Horn, have made me feel the same way. Finally, I would also like to acknowledge the support of my sisters, Aisha, Azra, Amna, Christy and Michelle; my brother Ahmad Z. Javaid; my cousins in the US, Zafar, Ishrat, Nusrat, Ehaab and Aymen; and some of my closest friends: Shahid, Tejas, Crystal, Sarim, Farid, Shahrukh, Atif, Masood, Shamsher and Rizwan.

Contents

List of Tables

List of Figures

List of Abbreviations

ADS-B	Automatic Dependent Surveillance - Broadcast
AHRS	Altitude and Heading Reference System
BLOS	Beyond Line of Sight
C3UV	Center for Collaborative Control of Unmanned Vehicles
DDoS	Distributed Denial of Service
ETSI	European Telecommunications Standards Institute
FAA	Federal Aviation Administration
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GSCS	Galileo Satellite Communication Simulator
GSSF	Galileo System Simulation Facility
LaBRI	Laboratoire Bordelais de Recherche en Informatique, Bordeaux University, France
LOS	Line of Sight
MSSE	Multi-Scale Satellite Simulation environment
MTBF	Mean Time Between Failures
NIST	National Institute of Standards and Technology
NORAD	North American Aerospace Defense Command
Open-SESSAME	Open-Source, Extensible Spacecraft Simulation and Modeling Environment Framework
OS3	Open Source Satellite Simulator

PNT ..... Position Navigation and Time

RAIM ..... Receiver Autonomous Integrity Monitoring
RTT ..... Round Trip Time

SCUAL ..... Swarm of Communicating UAVs at LaBRI
SNACS ..... Satellite Navigation Radio Channel Signal Simulator
SPEEDES ..... Synchronous Parallel Environment for Emulation and Discrete Event Simulation

TLE ..... Two-Line Element
TTL ..... Time To Live

UAV ..... Unmanned Aerial Vehicle
UAVNet ..... Unmanned Aerial Vehicular Network
UAVS ..... Unmanned Aerial Vehicle System

Chapter 1

Introduction

The remarkable growth in mobility-aware and location-aware vehicles, as well as related applications, has led to wide use of such vehicles in almost every domain. Most of these application domains put human lives at risk due to the unknown nature of terrain, weather or other environmental factors. One of the most popular class of vehicles under this category is unmanned aerial vehicles, UAVs. These systems can be sent to distant planetary bodies for research (e.g., Philae aircraft landed on a comet after a 10 year journey [1]) or to detect and survey real-time catastrophes like earthquake [2] and forest fires [3]. UAVs have found their use in applications like agricultural chemical deployment [4], ecological surveys [5], natural event monitoring (Hurricane Hunter [6]), disaster management [7] and humanitarian response (e.g., damage assessment, search and rescue operations, dropping relief supplies in case of emergency [8]), 3-D Mapping and photogrammetry [9], wildlife protection [10], etc. By 2018, only the military UAV market is expected to reach the $80 billion mark [11] while commercial UAV market is forecasted to reach close to $9 billion.

1.1 Background

In early 2015, commercial UAVs were authorized to fly in US National Airspace (NAS) by FAA and more than 7500 are expected to be seen in the air by 2020 [6]

compared to the number being negligible in 2014. These have also promoted research in industries and academia. Even today, FAA doesn't require people to obtain a license for drones (small and mini-UAVs) which are used for recreational purpose. However, they still limit drone usage up to a height of 400feet and away from the airport and air traffic [12]. Availability of low-cost mini-UAVs and DIY drones have also promoted recreational use as well as research. Nowadays, constructing a drone is possible in as less as $300 by purchasing separate parts from online stores [13]. Clearly, incorrect or failure in accurate navigation may lead to dangerous and life-threatening accidents for medium-sized UAVs and thus, necessitates prior testing of the model in a simulation environment.

1.1.1 Recent Attacks and Failures

With advancement in technology, the application domains of drones are no more limited to labs or defense. They can also be used by hobbyists, pranksters and troublemakers. This popularity may lead to an increase in the threats to the general public as well as chances of adverse usage of an increasingly cheaper technology. After Iran's claim of RQ-170 capture, an in-depth study of UAV vulnerabilities has been done by several researchers including our ACRL (Advanced Computing Research Lab) team at The University of Toledo. Through our studies, it was understood how easily a UAV can be compromised and attacked. In 2012, North Korea launched a GPS Jamming attack on its soil bordering South Korea, which disrupted the navigation of aircraft, ships and ground vehicles [14]. Several other works discuss recent attacks on UAVs [15]. A recent news in Washington Post detailed 47 biggest drone failures during $2001 - 2013$ and the plan of US DoD to extend UAV operations to 110 bases in 39 states by 2017 [16]. Figure 1-1 shows the location of these drone crash incidents of the severe category (called class A) and the Pentagon's extension plan of operation bases to 110.

It was noticed that until 2007, the number of reported cyber-attacks either in the civilian domain or to military systems were negligible compared to past few years. The primary reason behind the absence of attacks were the low popularity of these systems in the civilian domain, which didn't give adversaries much opportunity to study and exploit these systems. An earlier incident of satellite hacking was reported in 2007, which involved a British satellite being controlled by a terrorist group in 1999 [17]. In 2007, another news reported a US Satellite being used by LTTE (a Sri Lanka-based terrorist organization) to broadcast their messages and videos using some free bandwidth [17, 18].

The first major case of an attack on a UAV system was the discovery of the recording of a UAV feed when some members of an Iraq-based terrorist group were captured in December 2009. The video feed was captured using a $26 software called SkyGrabber which was designed to capture free satellite-based entertainment channels, using a satellite antenna [19]. This incident occured due to the reason that terrorists discovered the vulnerability of the video feed being unencrypted. It came to the light later that this vulnerability was known to the Pentagon since early 1990s [20]. In September, 2011, a malware was found in a Control Room computer of a USAF Base, which was serving as a base station for UAV Command and Control Network [21]. Later, it was declared as just a Keylogger, but clearly, was a huge threat to the national security. On the contrary, physical attacks on these drones pose a threat as loss of technology to adversaries and troublemakers. In December 2011, Iran claimed that it shot down a US $RQ-170$ stealth drone [22]. In 2014, they again claimed that they have created a copy of that captured drone through reverse engineering the UAV design [23]. In August 2014, Iran again claimed that they have shot down and captured an Israeli stealth, radar-evasive type drone called Heron [24] which could be further reverse engineered and used against the US and its allies.

Through this discussion, we understand that it is important to evaluate the risk

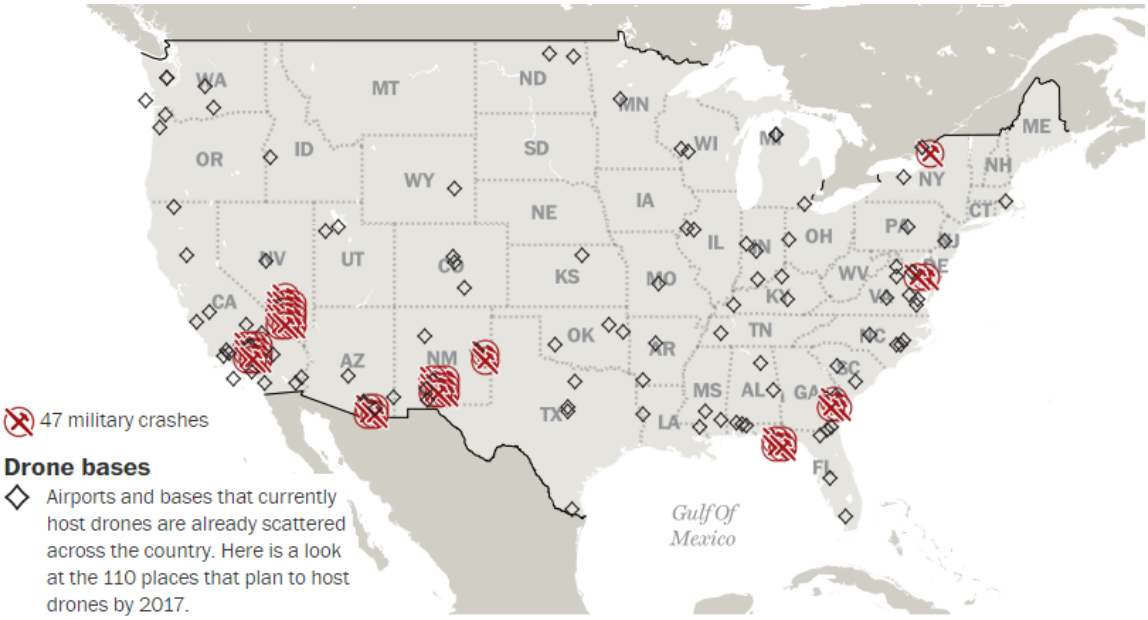

Figure 1-1: Class A UAV crash sites (2001-2013) and planned expansion of drone operations to 110 bases [16].

and vulnerability of a UAV for cyber-attacks based on its components including its comm-links, storage units, fault handling mechanisms, etc. A report published in 2001 lists the causes of failure for UAVs as "insufficient testing before purchase" and suggests that it is common to all failed programs [25]. A lot of money invested in these programs also went waste. Availability of a proper testbed to test these systems before the flight would lead to the prevention of midair collision and ground casualty. It would also help prevent any loss of human life and minimize investments in failed experiments.

Recent increase in attack attempts on these unmanned systems have raised concern among defense as well as commercial manufacturers. With increasing autonomy level of these systems, concerns over their use have been ever increasing. These concerns necessitate the need of cost-effective and safe virtual simulation testbed environment for testing the accuracy of various security implementations in an Unmanned Aerial System (UAS). Addressing various environment variations, loss of connectivity,

and contested communication are some of the important aspects of such a simulation testbed due to the dependency of UAVS security on communication.

1.1.2 Motivation

This dissertation focuses on design, development and performance evaluation of a simulation testbed at the Advanced Computing Research Laboratory (ACRL) of the University of Toledo, so that various cyber-attacks can be implemented and simulated. Although UAV development started in early 1960s, the primary objective of the research has been its mission accomplishment capability, reliability, and efficiency in terms of time and power. Not much attention has been paid to the cyber-security aspect of such systems until recently. The most important issues in this area are - vulnerability, breach and threat identification; and corresponding attack prevention, mitigation, and recovery. Most works in this specific area focus on the causes and methods of security breaches at the lower-level system components. Surprisingly, very few works focus on the application or communication security of these unmanned systems. The need of a simulation testbed which can simulate the single or multi-UAV behavior and provide a realistic response in case of an attack served as our initial motivation. It is clear that navigation is one of the vital aspects of unmanned systems. Therefore, its availability is significant. The auxiliary goal of providing the academia with a cost-effective mode of simulations was also accomplished simultaneously.

1.2 UAV Network Architecture

This section discusses the unique nature of a UAV communication network and how it is different from other types of networks. Further, we discuss different types of UAV network (UAVNet) architectures used during UAV operations and missions.

From a security and threat analysis perspective, it is necessary to understand

that a typical UAVNet is different from the traditional computer network. Some researchers have compared it to wireless sensor networks (WSNs) [26] and mobile ad-hoc networks (MANETs). Although UAVNet bears a close resemblance to WSNs with respect to use wireless communication protocols [27], there are other aspects in which they differ. For instance, power requirements, the amount of information being carried by channels, and the number of nodes in a WSN are much lower than in a UAVNet. Moreover, the coverage area for a UAVNet is almost a thousand times bigger than that of a WSN. To reduce power consumption, all nodes in an WSN transmit their sensor data to a central node, which communicates with external systems. On the contrary, in the UAVNet, multiple architectures are possible. Some researchers have also combined the application of UAVs in sensor networks so as to utilize the bigger coverage area of UAVs [28].

Tactical UAV networks need to follow pre-defined communication architecture to complete complex operations. They also need to address issues like swarm co-operation and unforeseen disturbances. In the following section, we discuss some widely accepted and followed communication network architectures for UAVNets [29]. Broadly, these can be classified into two categories: Centralized and Decentralized.

1.2.1 Centralized

In this kind of UAVNet, one GCS/MCS (Ground/Mission Control Center) is connected to a single UAV called Master UAV (MUAV) and serves as its command center. All other UAVs are connected to the single GCS/MCS through this MUAV. All other UAVs use LOS links to communicate with MUAV or each other, and any UAV can connect to the Satellite due to the temporal as well as the spatial omnipresence of communication satellites. Figure 1-2 shows such a network.

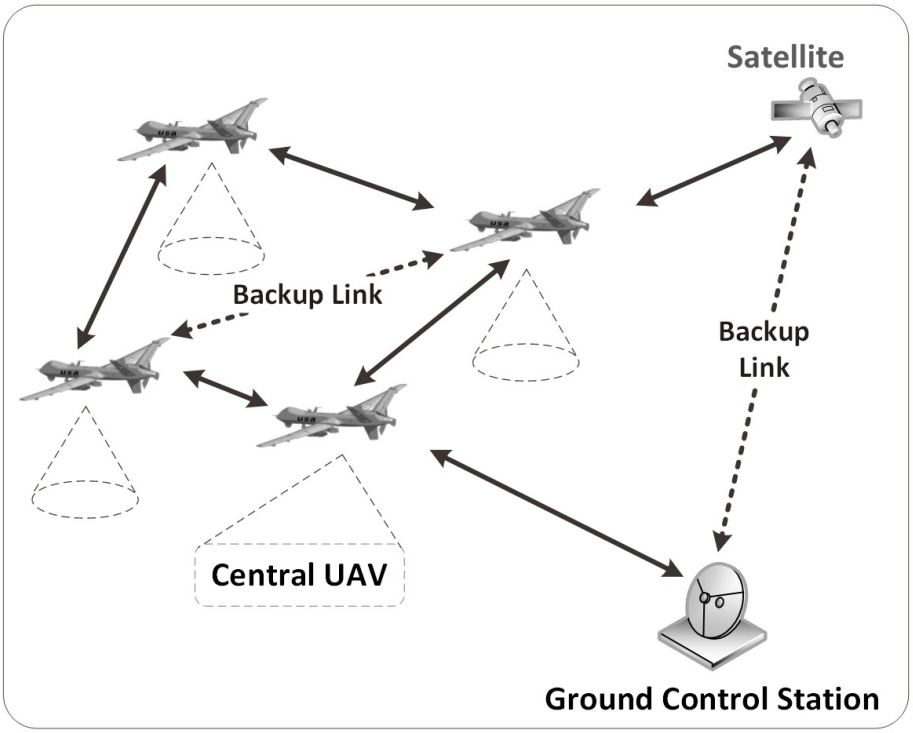

Figure 1-2: Centralized UAV communication architecture showing the central UAV and backup link

1.2.2 Decentralized

In this type of communication architecture, several GCS/MCS and MoCS (Mobile Control Centers, e.g., Laptops, PDAs) may be used with several UAVs communicating to them. The UAVs may use LOS or BLOS links. BLOS links may be established through other UAVs or Satellites. There are three sub-types of the decentralized UAVNet architecture based on the number of UAVs indirectly communicating with each of the control stations and are discussed below:

1.2.2.1 Single UAVs

Single UAVs directly communicate to the control stations without using other UAVs as repeaters. As a backup, the UAVs can still use other UAVs or a satellite in case they are too far away from the control station or LOS communication is not

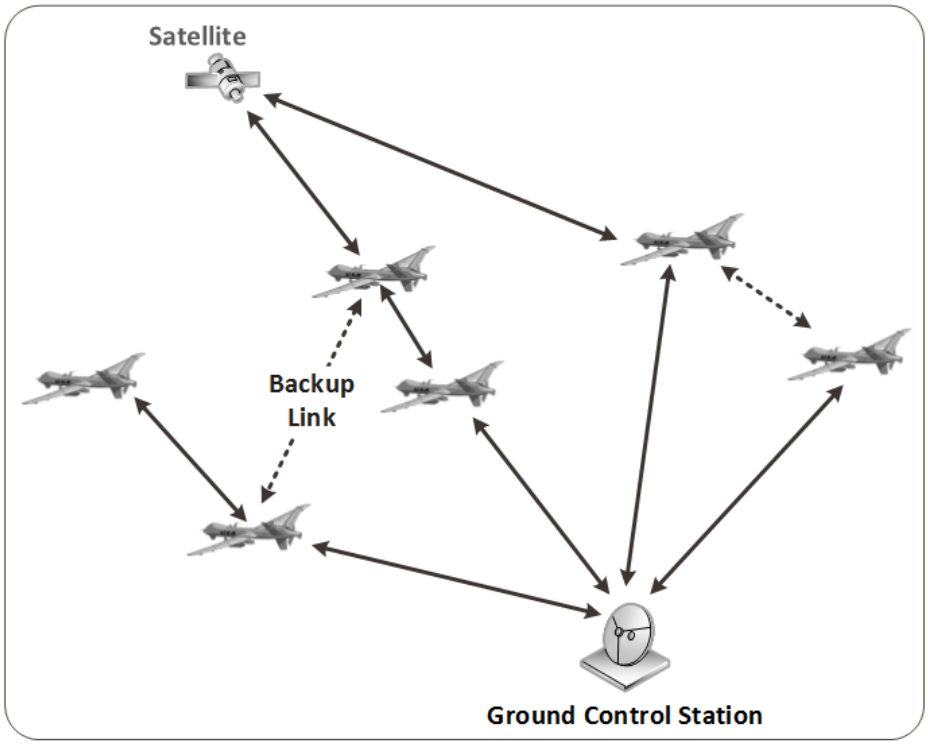

Figure 1-3: Decentralized UAV communication architecture

possible. Figure 1-3 shows such a UAV Network scenario.

1.2.2.2 Swarms

In this scenario, multiple UAVs communicate with a single control center, forming groups or swarms. The UAVs may also have connectivity with each other through LOS or BLOS links. There can be multiple swarms connected to a single GCS as well. Alternatively, it is also possible that one swarm can serve as a Master Swarm and other swarms connect to the GCS through this master swarm.

1.2.2.3 Mixed

In this type of communication architecture, multiple UAVs communicate with several GCS, UAVs and Satellites. Traditional networking topologies may be deployed but usually, such decentralized UAVNet have on-the-go communication establishment

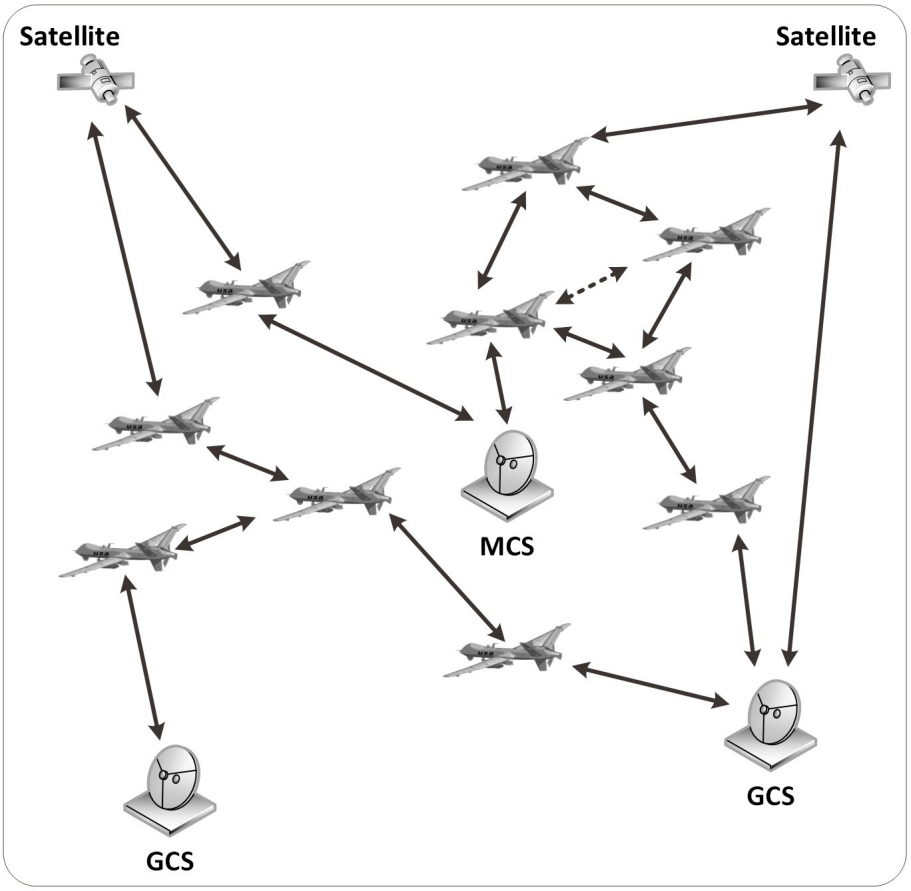

Figure 1-4: Multiple Swarms in a Decentralized UAV communication architecture

according to the assigned mission. Figure 1-4 shows an example of such a network.

1.3 Related Work

This section discusses various works that have been done in the area of UAV Network simulation. Many works deemed out of scope, as they primarily focus on modeling of a single UAV. These works mostly attempted to improve the system performance, flight range, or usability, in a closed laboratory or an open but controlled environment. It could easily be concluded that there is a lack of close-to-real simulator for UAV Networks (UAVNet), including UAVs, Satellites, and Ground Control Sta-

tions (GCS), in academia. An ideal testbed should allow inter and intra-component communication, and component level behavioral analysis. We present a classification of these existing simulators based on their capability to simulate multiple and single UAVs as well as the presence of navigation module in the simulation environment.

1.3.1 Software based Single UAV Simulation

Software simulation testbeds are mostly based on well-known software platforms and do not employ any hardware. Testbeds developed using Matlab-Simulink [30], FlightGear [31], JSBSim-FlightGear [32, 33] and Matlab-FlightGear [34] are some of the recent attempts of developing simulator for UAVs. All these simulation testbeds have focused on testing a single-UAV model instead of modeling its behavior in the presence of other UAVs in the real world.

1.3.2 Software-Hardware based Single UAV Simulation

Some other simulation testbeds using hardware along with software, have also been developed where the hardware might be actual UAVs [35], robots [36, 37], or just laptops [38, 39]. A very recent work of this type [33] focuses on analytical, and component-based simulation and analysis. This work attempts to address possible cyber attacks resulting in sensor compromise of various degrees.

1.3.3 Software based Multiple UAV Simulation

This type of simulation testbeds are also based solely on software platforms and are developed in-house. SPEEDES, Synchronous Parallel Environment for Emulation and Discrete Event Simulation [40], was found to be a promising simulator in this class. SPEEDES has the capability of simulating a swarm of UAVs on a high-performance parallel computer to match the communication rate and speed of a real

UAVNet. Another recent work of this class, DCAS (distributed cyber attack simulator), presents a distributed simulation framework for modeling cyber attacks and the evaluation of security measures [41]. DCAS is based on an open source HLA (high-level architecture) simulation engine called Portico. DCAS was designed for generic wireless and wired networks and can't use mobile components. UAVSim addresses these capability limitations by incorporating mobile components and related mobility models, radio propagation models, ad-hoc routing protocols, etc.

1.3.4 Software-Hardware based Multiple UAV Simulation

This type of simulation testbeds use a combination of software and hardware platforms, and real or emulated UAVs can be plugged into the simulator. These are developed in-house for UAV related research, instead of being commercial. In this category, *C3UV*, Center for Collaborative Control of Unmanned Vehicles at UC Berkeley [42] and SCUAL, Swarm of Communicating UAVs at LaBRI [43], were found to be most promising. Since 2004, the *C3UV* team has continually updated their simulation environment. In 2013, the capability of multiple-UAV simulation on high-performance parallel computing environment was developed. *C3UV* allows the use of real UAVs interfaced to the simulator. On the other hand, SCUAL allows use of real mini-UAVs in a restricted area at the LaBRI, Bordeaux University, France. SCUAL can be used by other researchers by permission and with some restrictions. Despite all their achievements, *C3UV* and SCUAL involves enormous expenses in terms of the high-performance parallel computing systems or hardware in terms of real UAVS. UAVSim, on the other hand, was designed to be cost-effective, and the extra investment is only required if the tested mechanism needs to be implemented on a UAV.

1.3.5 UAV Simulators with GPS

A visual $3D$ flight simulation software based on Matlab and Simulink was an early attempt towards simulating UAV that used the navigation module of FlightGear [34]. FlightGear provides a generic GPS support with GPS receivers yet to be implemented [44]. Another project called UAV Playground was developed in Java used FlightGear to receive GPS data and achieved GPS tracking through Google Earth [45]. In industry, the aeronautical division of IDS Corporation has designed an unmanned aerial vehicle simulator Hero UAVSim [46] composed of ground control stations (GCS) [47], UAV Simulator, and a sensor payload simulator [48]. The UAV Simulator has GPS-based auto tracking capability and GPS outage mitigation measures while GCS has an integrated GPS receiver to determine its actual position.

1.3.6 Independent Hardware based GPS Simulators

Several hardware based GNSS simulators are available for purchase from various manufacturers. LabSat simulator [49], a low-cost hardware based simulator, provides selection options as GPS, GLONASS, Beidou, and Galileo. It generates genuine navigation signal that can be stored, replayed and used in different applications. Spirent implements several similar hardware, e.g. GSS9000 multi-frequency, multi-GNSS RF constellation simulator for professional, controllable and repeatable testing in the lab [50]. A GPS simulator by National Instruments (NI) can produce GPS signals of up to 12 satellites and are aimed at testing GPS receivers [51]. Many other simulators such as IFEN Inc. NavX-NCS Professional/Essential, CAST Navigation SGX GPS Satellite Simulator, AeroFlex Portable GPS/Galileo/SBAS Positional Simulator GPSG-1000, etc., also simulate GPS and GNSS but differ with respect to the range of signals produced and constellation implemented. In academics, researchers tend to incline towards open source solutions rather than expensive hardware based solu-

Table 1.1: Comparison of software based GPS Simulators

Simulator	**GNSS Type**	**Language**	**Visualization**	**Vehicular Network Simulation Capability**
GSSF	Galileo, GPS	C++	Yes	No
SNACS	Supports All	C++ and MATLAB	Yes	No
Open-SESSAME	Supports All	C++	Yes	No

tions. All the devices discussed here are quite expensive and could not be considered as low-cost for a UAV simulation testbed developed in an academic setup.

1.3.7 Independent Software based GPS Simulators

Very few researchers have developed pure software-based GNSS simulators. There were few simulators available for purchase too, but our focus lies in open source solutions. GSSF, SNACS, and Open-SESSAME were the top-three in this category. Table 1.1 gives a brief, comparative analysis of these three software-based GNSS simulators. Galileo System Simulation Facility (GSSF) [52] focuses on the Galileo navigation system. Implemented in C++, it allows longer simulations and large geographical area coverage. It provides raw Galileo and GPS signal generation, express-mode simulation, and enough functionality to analyze and visualize data. Satellite Navigation Radio Channel Signal Simulator (SNACS) [53], on the other hand, is a single GNSS satellite signal generator. It is open source and implemented in C++ with parallel processing. The radio channel input of SNACS and simulation results can be analyzed in MATLAB. Open-SESSAME (Open-Source, Extensible Spacecraft Simulation and Modeling Environment framework) [54] is another popular simulator that provides dynamics simulation for spacecrafts for developing hardware as well as testing flight algorithms. Based on C++, it not only provides attitude and orbit

modeling, but can also be applied for orbit simulation, space environment assessment or control algorithm validation. One of the biggest limitations of all these simulators was the unavailability of any interfacing with any network simulation software to enable vehicular-network simulation.

1.4 Research Objectives and Contributions

This dissertation primarily focuses on the design, development and testing of a simulation framework for the location and mobility-aware unmanned aerial vehicular networks (UAVNets). As a prerequisite, detailed threat modeling and risk evaluation has also been performed and discussed in this work. Although more detailed requirements of such a testbed have been discussed in the Chapter 3, it should be clear that such a framework needs to allow any UAVNet simulation including its various systems, such as navigation. In this work, we demonstrate the navigation capability using GPS. Therefore, all the discussion from here onwards may refer to GPS and GNSS interchangeably. This simulation framework, called UAVSim, addresses various security simulation requirements as well [55]. Additionally, the GNSS/GPS simulation framework have been independently designed so that it can be used in UAVSim as well as with other OMNeT++ based vehicular network simulations. The outcome of this research is expected to solve the problem of accurate security simulations of different types of UAVs available in various domains.

To sum up, following technical contributions have been made in this dissertation:

- A detailed UAV, as well as the UAV system architecture, was developed through extensive literature survey for accurate system description. In the overall process of framework development, this served as a prerequisite to the threat modeling phase.

- Detailed Threat model using the CIA triad and Risk evaluation using standard

risk evaluation grid to identify highest priority threat. Once a testbed was developed, this model along with detailed system description provided

  - enough details to specify requirements and define the design of the simulation framework
  - list of attacks that need to be implemented and understanding of their detailed anatomy for implementation

- An OMNeT++ based open-source simulation framework for security as well as swarm simulations of UAV Networks with the flexibility of each and every environment parameter variation including navigation, weather, mission path, transmission range, etc. The functionality has been tested and demonstrated through implementation of various attacks as well as some defense mechanisms.

- An OMNeT++ based independent navigation simulation framework, incorporating satellites and weather data, has been developed. It enables users to capture the characteristics of unmanned aerial vehicle mobility and address the complexity of atmospheric pathways.

- Both the Satellite-end GPS signal transmission and vehicle-end GPS signal receiver have been implemented using standard trilateration equations. This receiver can be used in OMNeT++ for simulation as a component of any node, i.e., it can be used in other vehicular or mobile network simulations to incorporate GPS navigation.

- An extensive simulation study has been performed to reveal the achievable performance of the simulation framework using a broad range of affecting factors. Most importantly, the impact of the number of concurrent users, mode of simulation, etc. has been investigated. These studies prove the correctness and reasonable performance of the developed framework.

1.5 Dissertation Outline

Chapter 1 introduces the dissertation topic, its background, and the motivation behind it. It also discusses related work done in academia and finally, describes various research objectives and the major contribution of this work.

Chapter 2 presents detailed threat modeling and risk evaluation. It also emphasizes on command and control hierarchy definition for UAV systems being used throughout the US. The threat modeling follows a goal oriented approach and uses the CIA triad, and risk evaluation is based on standard risk evaluation grid.

Chapter 3 explains the detailed design and requirement of the developed simulation testbed framework. It also discusses platform evaluation and features of some of them. Next, the implementation of the simulation framework along with detailed description of all the modules are also discussed in this chapter.

Chapter 4 describes the anatomy and implementation details of various attacks identified during the threat modeling phase. This chapter describes the working of these attacks and how some defenses have been implemented. It also investigates the performance of testbed for high volume traffic during the attack and non-attack simulations.

Chapter 5 presents various performance evaluation results. This evaluation focuses on gauging whether the testbed is performing decent enough so that it can be used by other researchers with generic computing infrastructure. Tests were performed, and trends were plotted for various attack simulations in this chapter.

Chapter 6 concludes the dissertation by summarizing major results and findings obtained in this research and gives recommendations for future work. It discusses possible extensions of some modules and what kind of attacks and their mitigation/detection measures could be implemented.

Chapter 2

Threat Modeling and Risk Evaluation

Much of the research related to security and threat modeling arise from a low-level system perspective and mostly, focuses on the causes and methods of computer security breaches. Nonetheless, these works are focused on answering the same questions: what are the vulnerabilities of the system in question, how can attacks be prevented and how can the threats be mitigated. Possibly, a better approach is to perform a cause-effect analysis defining the cause of unintended or degraded subsystem functionality and evaluating the attack severity on mission/task performance [56]. This served as the primary motivation for the proposed threat analysis and modeling before the design and development of the introduced simulation testbed framework UAVSim.

Figure 2-1 shows the typical UAV Network communication scenario, which consists of several components and different kinds of communication links. These links carry different types of information and data [57]. Typically, this type of network has three types of links based on the information being transmitted, namely, UAV to the base, UAV to UAV and UAV to Satellite link. UAV to base links carry telemetry data, audio, video, and control information. Besides this, satellite links carry GPS, weather, and meteorological information. UAV to UAV links are usually used in case

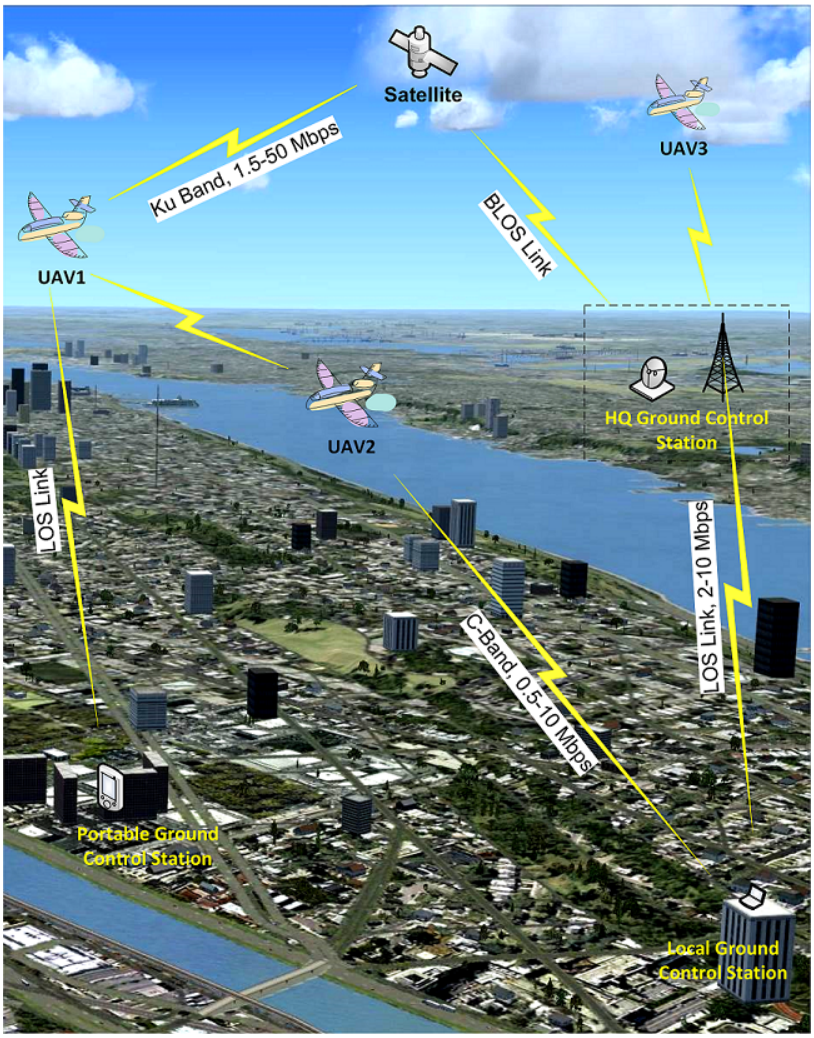

Figure 2-1: Typical UAV network communication scenario

of emergency of lost satellite and base communication link and are used as emergency channels to transmit any required information.

One of the vital uses of UAVs is in ballistic missile defense networks where UAVs are typically tasked to patrol an intermediate area between the ballistic missile launch site and the missile's intended target. Ballistic missiles are capable of moving at an extremely high speed and require quick detection, tracking, and elimination. Particularly, detection and tracking should be done as quickly as possible after the launch to increase the chances of successful interception. The ballistic missile defense network

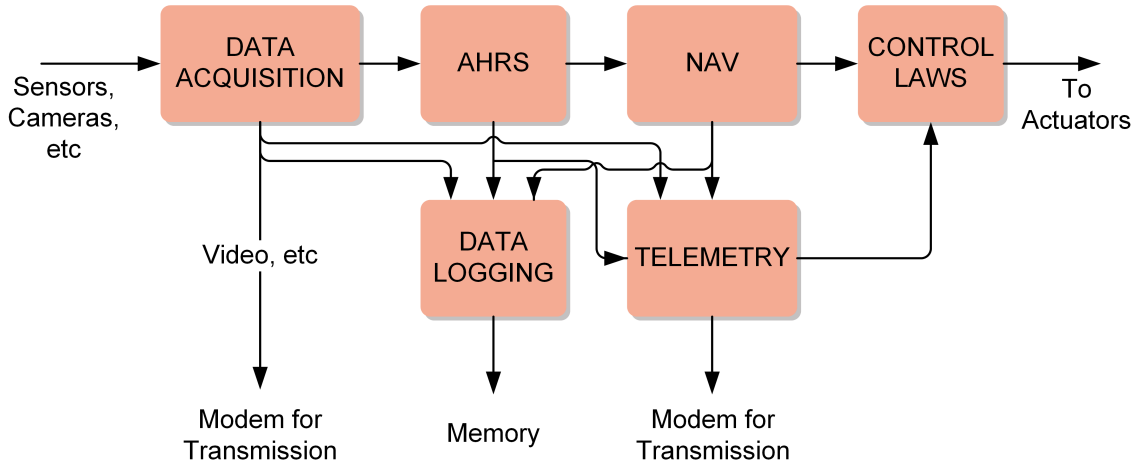

Figure 2-2: UAV architecture with basic components

designers intend to intercept the missiles during the boost phase that typically lasts 2-5 minutes. During this time, the missile may quickly move out of the range of UAV sensors if they are flying directly away from the UAV flight path. Therefore, information routing in the ballistic missile sensor network has to comply with high availability as well as security requirements [58]. This case was discussed here to illustrate the time sensitive nature of applications related to UAV and the importance of communication channel security. UAVs of the size of a hummingbird have also been designed and operational [59], which clearly indicates that there are no size or portability problems with these cyber-physical systems.

2.1 UAV System Modeling

2.1.1 Single UAV Architecture

As an abstract representation, any device or vehicle is usually represented as a black box during simulations to simplify analysis with respect to external factors. A similar approach was used for the initial UAV Model design. For simplicity, UAV

was abstracted as a node that can communicate using short distance as well as long distance radio signals, with nearby UAVs and satellites. This abstraction also resulted in faster computation and quicker simulation results.

Further, an advanced UAV model was developed to allow evaluation of more sophisticated component level attacks in a UAV. Figure 2-2 shows this advanced model detailing various system components. This model can be defined as a combination of six separate, but dependent systems. These modules are Data acquisition module, AHRS (Altitude and Heading Reference System), NAV (Navigation) System, Control Module, Data Logging Module and the Telemetry Module [60–62]. The communication system module has not been shown here as it encompasses all the modules and any incoming/outgoing command and control signals, and data signals pass through it. Primary functionality of these six modules are briefly discussed below:

- *Data acquisition module*: This module is tasked to collect data from the environment. It connects to various sensors installed on the UAV including camera, heat sensor and infrared sensor. It then sends the required information to respective modules. For example, attitude information is sent to the AHRS module while camera data is sent to the Telemetry module for further transmission. Any kind of dysfunction in this module could result in a catastrophic damage to the UAV.

- *AHRS*: An AHRS provides $3D$ orientation of an aircraft by integrating MEMS based gyroscopes on a circuit board and fusing this data with accelerometer and magnetometer data. With sensor fusion, drift from the gyroscopic integration is compensated through the use of reference vectors of gravity and the earth's magnetic field. AHRS is more cost effective solution than conventional high-grade IMUs (Inertial Measurement Units) that only relies on the high bias stability of the gyroscopes.

- *Navigation System (NAV)*: It is known that Navigation systems primarily collects and provides location information. Additionally, for systems like GPS, NAV system also collects timing and mobility information through highly accurate atomic clocks available on the Navigational satellites. These data are called PNT (Position, Navigation and Timing) and are very crucial for UAV operation and successful mission completion.

- *Control Module*: The control module connects to the actuators of the aircraft and handles sending various control signals. These control signals instruct the aircraft to move in a particular direction with a set speed and a specific orientation. Since most UAVs are remotely piloted, this module serves as one of the most important modules as any discrepancy in this module is unacceptable.

- *Data Logging Module*: All the data of any aircraft is usually logged in this module for checks during flight and further analysis in case of crash or system malfunction. This module stores log information of all data collection, reception, and transmission as well as stores PNT data after fixed intervals to keep track of mission.

- *Telemetry Module*: This module is responsible for automated communication and data delivery by the UAV to control station or to the satellite. Some measurements are also made within this module using data collected from the remote or inaccessible location by the Data Acquisition module. Data is finally transmitted to receiving equipment at control station for monitoring, either directly or through a satellite link, depending on link availability.

More about this detailed model and how these six modules have been implemented in our simulation testbed has been discussed in Chapter 3.

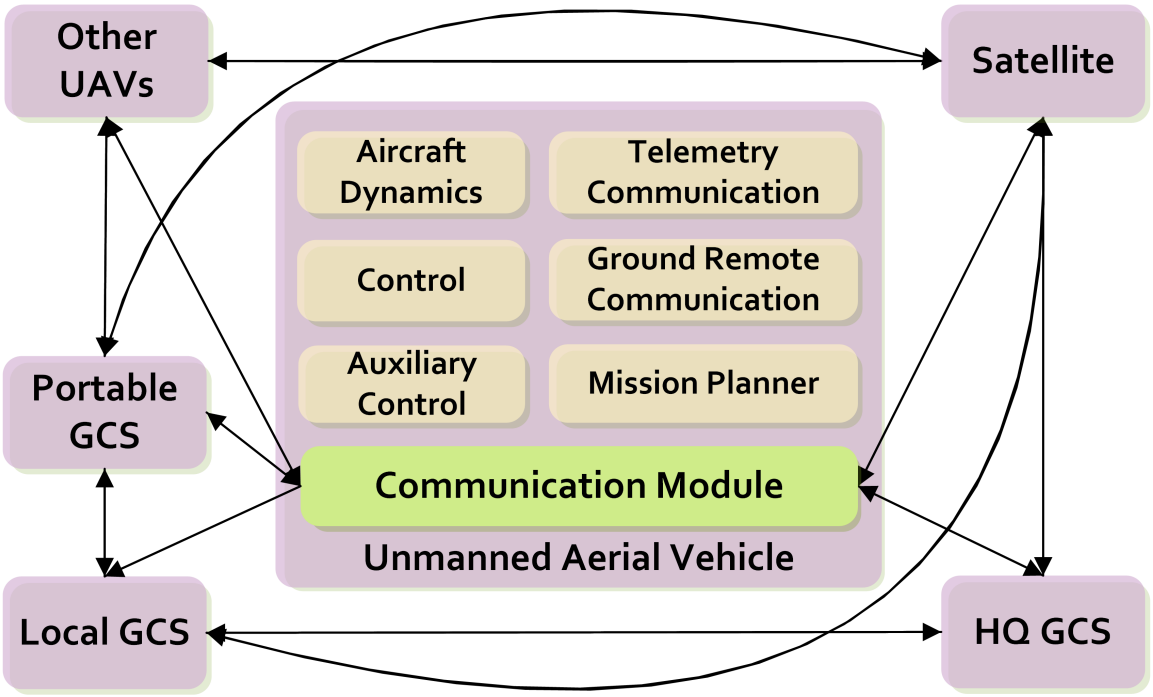

Figure 2-3: Simple block diagram representation of a UAV network

2.1.2 UAV Communication Architecture

It is clear that all communication channels used in UAVNets are wireless. This renders the communication channel vulnerable to various exploits and malicious attacks from aerial intruders flying in the vicinity as well as ground based adversaries. We have tried to include all communication channels for the system which are important with respect to communication and security. It is clear from Figure 2-3 that the different components of the system rely on wireless communication channels for communicating with each other. GCS (Ground Control Stations) can be of three types, portable, local and HQ. HQ GCS is usually located at a Command & Control Center or Base of the concerned Agency/Department. Local GCS are the mobile control stations in the war-zone, typically in the form of a radio-equipped vehicle. Portable GCS is a sub-class of local GCS and could be PDAs (Portable Digital Assistants),

smartphones or rugged laptops.

Although various communication channels in a UAVNet seem similar because of their wireless nature, there are huge differences among them with respect to security [63]. The link between a Satellite and a UAV is usually LOS (Line of Sight) radio communication. UAV-UAV, PDA-UAV, and Local GCS-UAV links could be LOS radio communication or GPRS/EDGE based using existing communications infrastructure. Studies show that by using BOTNETs, a botmaster, and a small drone, it is possible to break into existing networks through identifying vulnerabilities in these GPRS or Wi-Fi Networks. This vulnerability can be attributed to the insecure and unreliable security techniques used in Wi-Fi Networks [64]. Threats to each of these communication links and components are also different and have different security requirements. Components like Satellite and HQ GCS might have certain threats but may not be too vulnerable due to the existing security measures in place [65]. Further, the link between a UAV and the HQ-GCS might be BLOS (Beyond Line of Sight) in case the UAV is far away from the HQ-GCS. In such situations, a tower or satellite might be used as a repeater and thus introduces another security loophole.

2.2 Command and Control Hierarchy

One of the important aspects of understanding a system is being aware of the command and control hierarchy, especially when it is a complex system as UAVS. The complexity of these systems arises from the involvement of several agencies and departments in the US in independent drone operations hierarchy. At the top level, the system operations can be categorized as military and civil operations, as shown in Figure 2-4. Further, each operation type is carried out by several departments, and each one of them comprises of several agencies that have their specific missions, UAVs as well as satellites. This hierarchy is not comprehensive yet and is created

using information available in the public domain to give designers an idea of various stakeholders in commercial or military UAV operations.

Another aspect common in all UAV operations is approval and involvement of FAA. To this end, FAA strictly limits the use of UAVs up to a certain height so that other commercial and civilian aircraft are not endangered or affected. FAA does not require people to obtain a license for hobbyist drones (small and mini-UAVs) which are used for recreational purpose. Such usage is limited to a height of 400 feet and away from airport and air traffic [12]. Recently developed Lily Drone is a great example of such a drone that operates in preset flight modes, follows a tracker on the wrist of the user and lands near the tracker if it is low on battery [66]. As expected, even this drone goes only to a height of 50 feet. Understanding of these regulations and C&C hierarchy is also important for hobbyist and designers as these need to be part of the programming of the UAV.

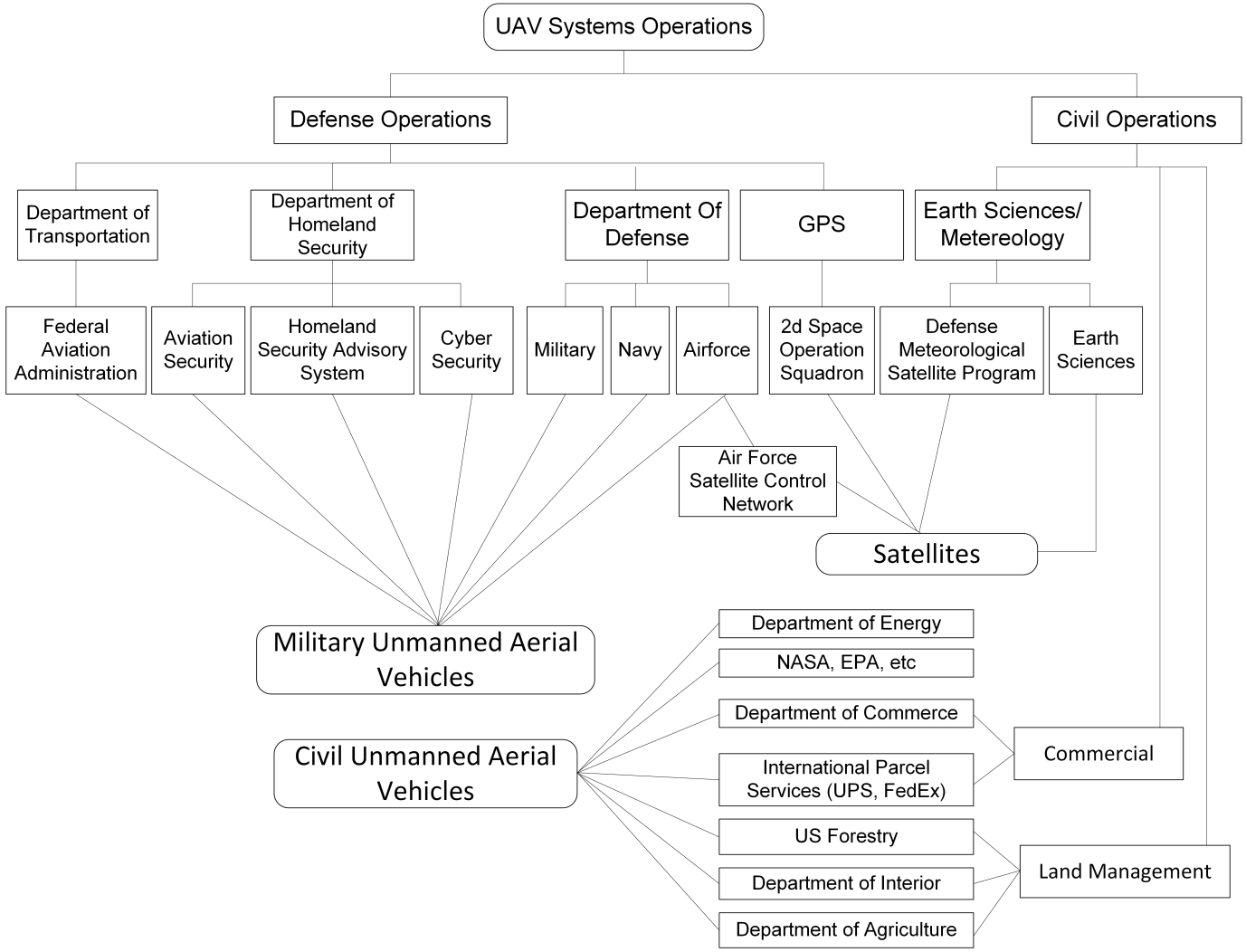

Figure 2-4: Command and control hierarchy of various UAVs in the US

2.3 Threat Analysis and Modeling

As mentioned earlier, threat analysis has been completed for the UAVS as a pre-requisite for designing a simulation testbed. By definition, threat modeling involves structured analysis of the security of the subject and enables designer to identify, quantify, and address the security risks associated with that subject. Consequently, it is considered an important aspect of ensuring the security of a system because it can lead to discovery of exploitable vulnerability in the system [67].

We follow a goal-oriented approach to the security threat modeling and analysis by using visual model elements to capture explicitly threat-related concepts [68]. Such a model enables system designers to understand the threat profile of a system through proper examination of the system as an adversary would. It also helps them to determine the number and types of high-level security risks posed to the system [69]. The resulting threat model describes potential attacks on the system. One of the uses might be to understand the attack severity and to evaluate decisions that will affect the security of the system over a long period of time [70]. The model can also be used as a basis for system penetration testing as the system evolves after several iterations of design, development, and testing [71].

Our attempt to model the threats posed to a UAV system and possible attack paths of these threats resulted in a threat model shown in Figure 2-5. We will now discuss these threats in detail using the developed Cyber-Security Threat Model.

2.3.1 Confidentiality Attacks

This property primarily deals with unauthorized access to information and the most common way of compromising this property is interception of information. The four major components of the UAV model which are vulnerable to this class of attack

are the UAV, GCS (all types), communication link and human beings. Threats to the GCS are mostly software based, namely, virus, malware, trojan, keylogger, etc. A major threat for a UAV is hacking. It should be understood that software based threats can affect UAV as well, but there are easier ways to get those threats to the UAV. The GCS security breach or the UAV security breach itself may lead to other threats to the UAV, but the need of addressing these threats is mostly fulfilled if addressed at the GCS level.

The primary source of compromising the security of communication links between various system components is through network attacks such as hacking, eavesdropping, identity spoofing, cross-layer attacks [72] and multi-protocol attacks [73]. Clearly, all these attacks might not apply to each of the links available in the system. The aim of including all of these attacks in one group is to identify possible threats to various communication links involved instead of identifying the threat for each type of link separately. When putting proper mitigation measures in place, it is required to notice which attacks affect these links and deploy the measures accordingly [74]. As for the human element, the increasing trend of online social and business networking has resulted in a rise of new kind of threats. Some of these include social engineering, fake online competitions, blackmailing and behavioral exploit.

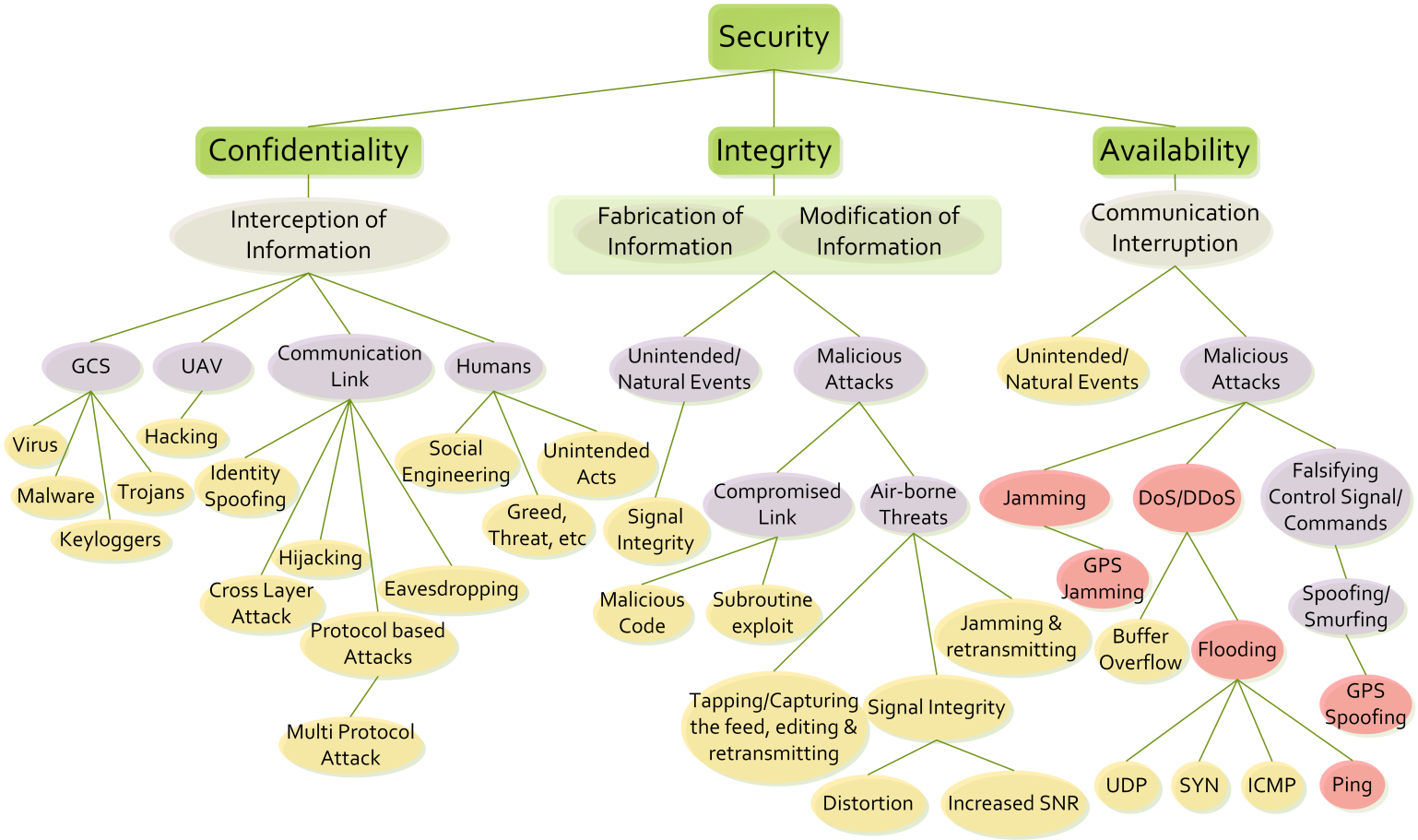

Figure 2-5: Threat model for an UAVS using the CIA triad

2.3.2 Integrity Attacks

The integrity of any system could easily be compromised using two basic operations, i.e., modification of existing information or fabrication of new information. Modification aims at altering the data during transit or in storage while fabrication involves creating new signals and transmitting it as if they were the original signals. Natural events like lightning, magnetic pole shifts, sun flares, etc., might cause some integrity loss and add unwanted noise to the signal. However, these natural events are occasional, and most communication protocols and equipment attempt to take care of issues caused by them by having an error detection and correction mechanism.

Next are airborne threats that have three broad categories: jamming, compromising the signal integrity and tapping or capturing the signal feed. Jamming aims at disrupting communication through interference with the reception. Researchers have proposed many defense strategies against jamming in wireless networks but even today, there is no effective solution to this attack. For compromising signal integrity, distortion and changing the SNR (signal to noise ratio) are the most common approach. The third way of tapping or capturing the feed is the most difficult type of attack to launch as it requires a lot of intelligence data in terms of transmission signal frequency, range, etc. Interestingly, a separate branch of communication engineering called Signal Intelligence involves the study of such attacks [75, 76]. The last class of attack in this category is fabricating or modifying information, which includes the use of malicious code or existing subroutines of the system. Subroutine exploit involves attacking the system through finding and exploiting vulnerabilities in the code of the system once the adversary has enough information about the system. This system information can be gained from a planned or brute force attack.

2.3.3 Availability Attacks

Primary cyber-attacks which might affect availability of the UAVS are jamming, falsifying signals and Denial of Service attacks (DoS). As discussed earlier, transmitting false commands or control signals requires a lot of signal intelligence and it can be a major threat to the UAV system availability. These false signals can actually make the UAV land or attack somewhere else through incorrect commands. DoS or DDoS (Distributed DoS) attacks are mainly based on network congestion or overflow in the network card of the system so that the system appears to be unavailable. During such an attack, the system or network is actually busy in serving other fake requests. Three ways of launching such an attack are flooding, spoofing or smurfing and buffer overflow. Flooding basically floods the network with one or more kinds of network packets by sending multiple packets to the system to be attacked. Usually SYN, UDP, ICMP and Ping packets are used in such an attack. Next type of attack belonging to this class is buffer overflow which aims at overflowing the buffer memory of network cards on the devices being used in the system. Smurfing involves flooding the system by broadcasting spoofed network packets and it seems to the target system that all packets are coming from different addresses [77, 78].

2.3.4 Threat Summary

Being a cyber-physical system, a UAS is vulnerable to most network oriented cyber-attacks but some of them can be more dangerous than others and can lead to a more unstable and vulnerable state of the UAS. Therefore, it is important to prioritize threats according to the risks they pose and their impact on the system once occurred. Based on this priority, threats should be addressed, and proper mitigation measures should be developed. We have presented a threat classification above, and it is published in [15] based on the initial analysis. Here, we present an updated and

Table 2.1: Major security threats to a UAS

Component/Technology	**Threat/Attack**
Satellite Communication	Weak Encryption
	Congestion due to Traffi
	Availability Attacks (Jamming, DoS, etc.)
Other Radio Comm Links	Compromised UAVs
	Eavesdropping
	Radio Jamming and DoS
	Location privacy attack
Unmanned Aerial Vehicle	GPS Spoofing
	Fuzzing Attack
	Hijacking and Immobilization
	Gain Scheduling Attack
Ground Control Station	Malware injection
	Keylogger and other data extraction mechanisms
	Weak Authentication
Command and Control Messages	Weak message authentication
	Control channel jamming
Sensing	Sensor and Actuator manipulation
	Spoofing

improved categorization of threats and associated attacks based on various components or technologies of the UAS. Table 2.1 summarizes these threats using several of our work [15, 55] and other attack reviews [32, 79, 80] in the literature.

2.4 Preliminary Visual Simulation

Initially, we focused on visual simulation of a single UAV so that impacts of an attack could be evaluated. We used FlightGear v2.6 and modified its inbuilt aircraft models to design a UAV model similar to the well-known Predator UAV. Figure 2-6

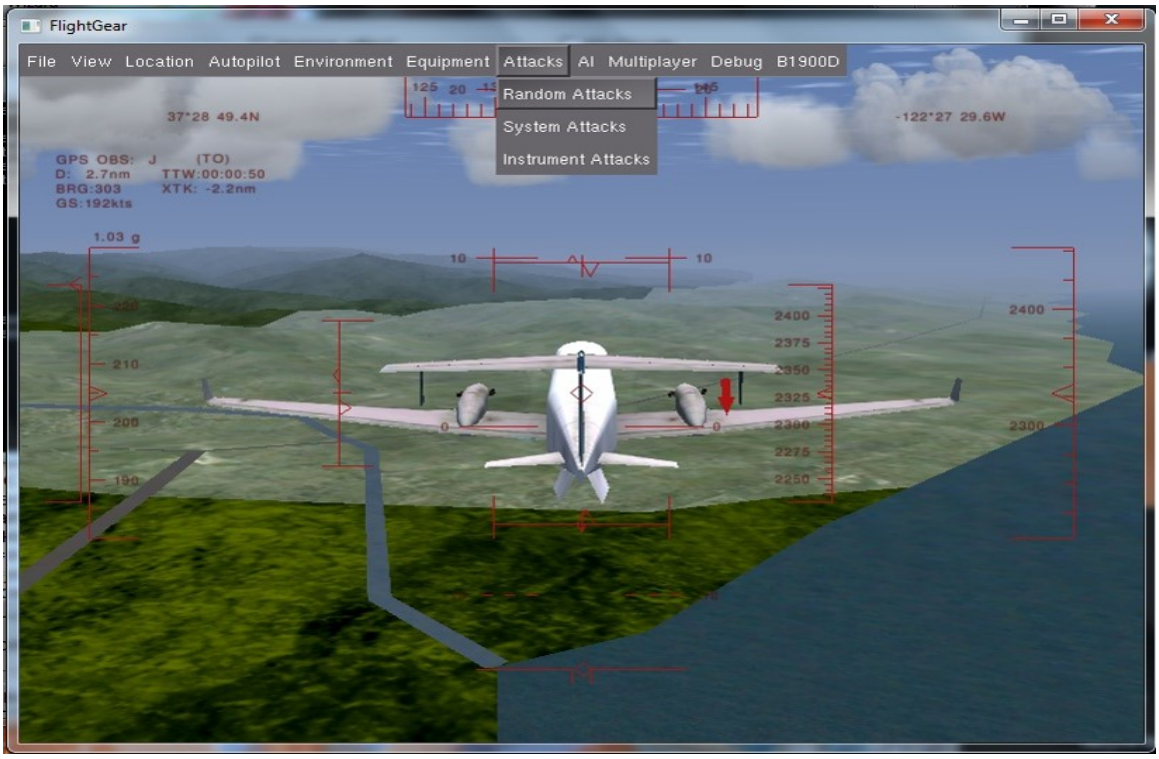

Figure 2-6: FlightGear simulation environment showing types of attacks introduced

represents the simulation environment of FlightGear showing the aircraft model in it. Different views are possible including the one shown in the figure. FlightGear is an open-source project that supports standard $3D$ model formats, and much of the simulator configuration is controlled through XML-based ASCII files. Writing 3rd party extensions for FlightGear (or even directly modifying the FlightGear source code) is simple due to its open source nature [81].

To simulate attacks in this visual simulation environment, we induced failures in various systems to check the system generated alerts and system response to those failures. Finally, we gauged the level of damage caused by failure to the UAV. It should be noted that the aircraft in these simulations were entirely dependent on the existing models designed using one of the three Flight Dynamics modeling software: JSBSim, YASim or UIUC (LaRCSim). Therefore, any alerts generated, failure mechanism

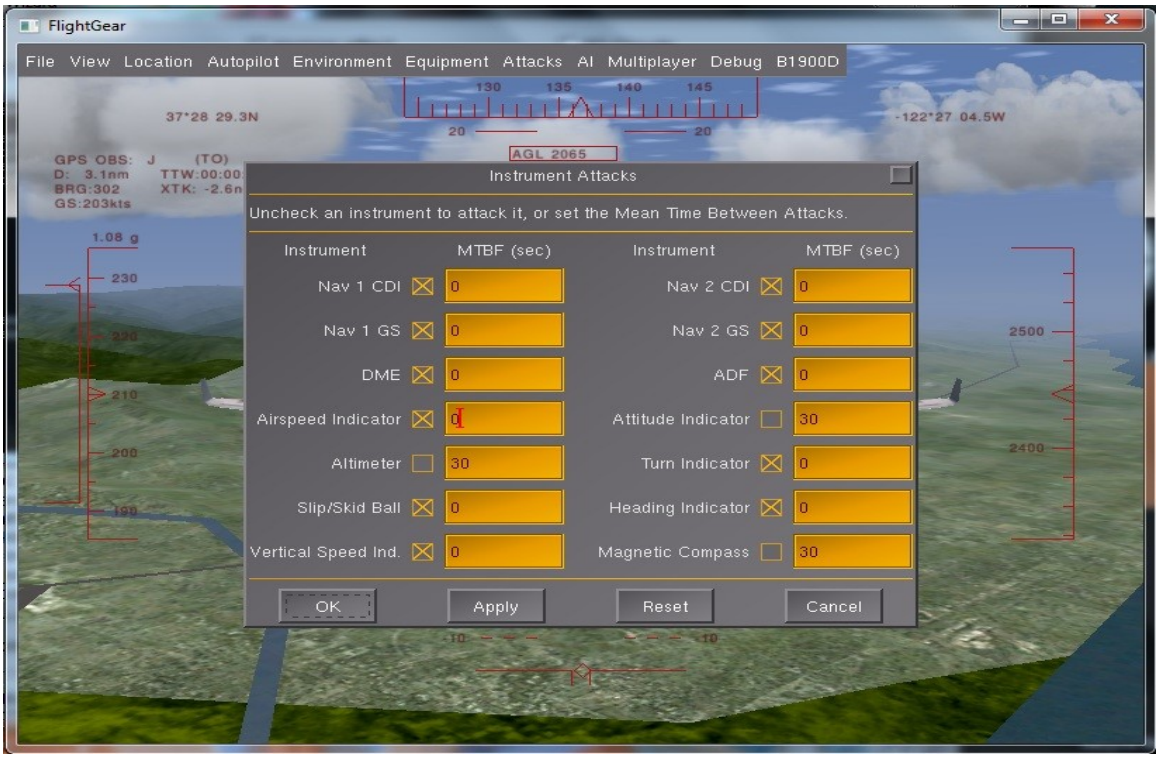

Figure 2-7: FilghtGear simulation environment showing various instrument attacks

activated, etc., should already be part of the model design. Otherwise, it is deemed to fail. Figure 2-7 shows various failures that could be inserted into the system as a result of cyber-attacks. These attacks can also be automated by specifying MTBF (Mean Time Between Failures) specifically for each and every kind of failure caused by an attack. Three types of attacks were introduced in this system as discussed here:

- *Instrument Attacks*: This category was used to cause failure of specific aircraft instruments including navigation, airspeed indicator, altimeter, vertical speed indicator, turn indicator, heading indicator, magnetic compass, etc.

- *System Attacks*: This category of attack was used to affect various aircraft systems such as, the vacuum system, pitot system, electrical system, all engines, aileron, elevator, rudder, landing gear, flaps, speedbrake, etc.

- *Random Attacks*: This category would cause failure on any of the systems as well as instruments on-board an aircraft and can be set using MTBF or mean cycles between failure (MCBF).

As a result of this visual simulation, it was noted that most system and instrument failure lead to a crash of the aircraft if it is in auto-pilot mode. One of the most important use of this simulation was in the next phase of Risk and Impact Evaluation where impact and the likelihood of various threats were defined as several levels. Finally, the risk posed by those threats were also calculated.

2.5 Risk Evaluation and Analysis

This section discusses the detailed risk evaluation and analysis posed by major threats using their likelihood and impact factors. We present the standard Risk Evaluation Grid used by the NIST as well as European Communication Standard Institute. Further, we discuss more on likelihood and impact factor evaluation and finally, we present results obtained by the detailed Risk Analysis.

2.5.1 Risk Evaluation Grid

Table 2.2 shows the Risk Evaluation Grid, which was used for the threat analysis of various threats. It has been defined as a standard grid in the ETSI (European Telecommunications Standards Institute) threat assessment methodology [82]. This grid is used for various detailed security analysis of European telecommunication systems, and a version of it has also been recognized by NIST for use in the US. This detailed system analysis helps the designers in determining the likelihood, impact and risk of each of the possible threats. It also gives developers an insight in the overall damage that can happen to the system due to a particular threat.

Table 2.2: Risk evaluation grid

		Rationale		
Criteria	**Cases**	**Difficulty**	**Motivation**	**Ranks**
Likelihood	Unlikely	Strong	Low	1
	Possibly	Solvable	Reasonable	2
	Likely	Strong	High	3
		User	**System**	
Impact	Low	Annoyance	Very Limited Outage	1
	Medium	Loss of Service (LoS)	Limited Outage	2
	High	Long time LoS	Long Time Outage	3
Risk	Minor	No need for countermeasures		1, 2
	Major	Threat need to be handled		3, 4
	Critical	High Priority		6, 9

2.5.2 Likelihood and Impact

Several researchers have tried to devise mechanisms for deciding the likelihood of various attacks in different scenarios, and it was found that the likelihood of attacks on various kinds of networks was entirely different. The likelihood of attack also varies for each component of the network. A quantitative evaluation algorithm that estimates risk indices by layering based on the intruding process has also been proposed [83]. This evaluation algorithm also involves simulation of the network and evaluation of threats and their likelihood based on the simulated network and the algorithm. One such example is the Threat and Vulnerability Analysis of WiMax/802.16 [78]. The issue with this type of analysis is that the risk values may be different for different researchers according to the information available and level of analysis.

The Likelihood parameter evaluates the possibility of attacks being launched. It is *Unlikely* if a potential attacker has much less information and needs to resolve several technical difficulties, or if there is a low motivation. It is *Possible* if there are lesser or no technical problems or if there are several reasons for someone to launch an attack.

Table 2.3: Risk analysis summary

Threat	**Algorithm(s)**	**Likelihood**	**Impact**	**Risk**
Jamming		3	1	3
Scrambling/Distortion		2	1	2
Eavesdropping		3	2	6
Cross Layer Attacks		2	1	2
Multi-Protocol Attack		2	1	2
Social Engineering		2	2	4
Spoofing	Device List	3	3	9
	X.509 device Auth.	2	3	6
Command and Control Message Modification	No MAC	3	3	9
	SHA-1 MAC	2	3	6
	AES MAC	1	3	3
Data Traffic Modification	Without AES	3	1	3
	With AES	1	1	1
DoS on UAV/GCS	EAP/SHA-1/AES/MAC	3	3	9
Signal Integrity		3	2	6
Malicious Code, Subroutine Exploit		1	3	3
Virus, Malware, Trojans and Keyloggers		3	2	6

It is *Likely* if there are a high motivation and no problems in launching the attack.

Impact signifies the resulting state of the system after an attack. It is *Low* if the attack creates only low-level problems, and the problems created are usually reversible and repairable. It is *Medium* if the attack is directed to loss of service for a single user for a considerable amount of time or limited scope outage for a multi-user system. The impact is *High* if the attack directed to an individual user causes a loss of service for an extended or longer periods with many users being affected and possible law violations or financial losses. The Likelihood and Impact vary from one to three as shown in Table 2.2. For a given threat, the Risk is calculated as the product of the Impact and Likelihood values.

2.5.3 Results

The result of detailed risk analysis is shown in Table 2.3. Evidently, the likelihood of an attack decreases if there are any security measures in place while the impact remains the same. For risk values less than 3, no countermeasure is required. As for values of $3-4$, the threat ought to be taken seriously. Finally, if the value is $6-9$, then the threat is critical and should be addressed as a priority.

It has already been discussed that this type of analysis is subjective, and risk values may vary according to the detail of analysis and information available regarding the system. It can be seen from the table that a reduction in likelihood results in an overall decrease in the risk. It should also be noted from the analysis results that Spoofing, DDoS, and command/control message modification pose the greatest threat to such a system and should be addressed on a priority basis. Other major threats are eavesdropping, software based threats, and signal interruption. Figure 2-6 shows the GUI of the FlightGear simulation software which was used to simulate the effects of these attacks and the severity of damage (usually a crash of the UAV). Results from this analysis will be further used while deciding which attacks to be implemented first.

2.6 Chapter Summary

In this chapter, initially, the architecture of the overall system and the UAV was designed by identifying major components of the system as well as that of the UAV. Two different models were proposed for the UAV: a basic model for faster simulations and an advanced model for detailed component level analysis. Subsequently, detailed threat modeling was performed based on these architectures. Also, associated attacks and possible attack paths were also defined along with the definition of potential

threats in the detailed threat analysis of the system. A summary of this threat model was also presented to list most imminent and high-likelihood threats to the UAV system. Finally, using the basic visual simulation and analysis of how the aircraft behaves under certain failures caused by attacks, the likelihood of each attack and how different attacks would affect the system was analyzed. All these steps helped us in detailed risk analysis of the UAV system and came up with high priority threats. As discussed in further chapters, this would ease the process of selection of attacks for implementation and testing of our testbed.

Chapter 3

Simulation Testbed Development: UAVSim

After the Threat Modeling and detailed Risk Analysis, our team was ready to design and implement the simulation environment. During this phase of the work, we first analyzed and evaluated a few open source options for the testbed development. It is necessary to reuse existing code rather than recreating it from scratch when we are aiming to develop an open source software for research and academic use. In this chapter, we first discuss the requirements for such a simulation environment then look at possible options for use. Later, we discuss various constraints and limitations faced by our team before and during the development process. Finally, we present the detailed design of UAVSim.

3.1 Testbed Requirements

The testbed should fulfill various requirements to be used as a cost-effective method of simulating UAVs and related security aspects. These requirements along with some of the advantages of using a software based testbed are listed as follows:

- UAV operations are supported by several agencies and department and thus require simulation capability of different terrains, weather, etc. Therefore, such a simulation testbed should allow users to simulate different mission scenarios through the use of various mobility models/paths, terrains, weather, etc.

- The aim of developing a testbed for UAVNet is to enable testing and evaluation of various threats, attacks and their defenses using the basic and newly designed UAV models. Therefore, the testbed should allow the use of different kinds of UAV models irrespective of the scenario being modeled.

- Enhanced security might result in the use of additional software or hardware components in the UAV. The testbed should be able to test the UAV system performance with these enhanced security measures in place. For detailed security analysis, a feature to evaluate effects on individual system components could also be provided.

- Since the testbed will be used by various classes of users, it should provide an interactive and easy to use GUI and result analysis module. Various intuitive options may also be provided for customized result analysis as per user requirements.

- The UAV in itself acts as a network of components that communicate with each other. The testbed should allow testing of relevant attacks and their effects on these UAV components. The designed UAV model should also replicate the real-world component communication behavior.

- Most UAV simulation tools developed until now focus on specific application requirement and are not specifically suitable for security analysis due to the lack of communication protocol implementation, proper GUI, and result analysis modules.

- Physical testbed involves investment in expensive hardware for the UAV development. This cost can be eliminated by using a software based testbed that allows users to vary simulation speed during the simulation and gain on-the-go insights.

- Use of a high-performance computer ensures that the simulation is real-time and also allows users to slow down the simulation speed. Both of these features may facilitate gaining valuable insights during the simulation.

3.2 Platform Evaluation

A comparative study was performed among popular and widely used open source network simulators to understand their capabilities and then decide which platform should be used for our simulation testbed development. Some graphics based flight simulation platforms like FlightGear were also evaluated, but due to their limitation of single UAV simulation, our focus moved to assess network simulators such as NS-2 and OMNeT++. After detailed analysis and comparison, it was found that OMNeT++ was most suited for our requirements due to its better network animation and support of a standard language, C++. Additionally, the network definition language (NED) for OMNeT++ is much user-friendly than Tcl (used in NS-2). Table 3.1 briefly summarizes this comparison. It is clear that OMNeT++ is preferred over NS-2 because of user-friendly programming model environment, hierarchical model structure, ability to run large networks and independent co-design of experiments [84].

OMNeT++ is open-source, has an excellent network animation module and most importantly, supports mobile node simulation. It is a C++ based, modular, open architecture, discrete event network simulator with strong GUI support and an embeddable simulation kernel that allows us to make modifications and develop our custom modules [85]. Once the base simulation engine was chosen, we evaluated other

Table 3.1: Comparison of OMNeT++ and NS-2 simulators

	OMNeT++	**NS-2**
Programming model	C++ and NED	C++ and Tcl/Tk
Model management	Independent kernel model	Embedded kernel model
Model structure	Hierarchical model structure	Flat model structure
Debugging and tracing	Present	Third party modules
Running large networks	Limited by system memory	Has scalability issues
Experiment design	Topology and models separate	All in one script

reusable modules that can be used for the testbed development. Two such modules are INET and OS3 (Open Source Satellite Simulator) [86]. INET is a built-in module of OMNeT++, which supports all kinds of wireless communication protocols, mobility models, and radio propagation models. OS3 is another OMNET++ based third party simulation module for evaluating satellite communication protocol. It provides a generic satellite constellation that seamlessly integrates real satellite tracks and weather data to simulate different conditions along with good visualization. Implementation of a highly accurate and stable satellite movement and modeling in OS3 provided a good base for the development of our navigation system. Being platform independent, OS3 can be employed easily on any system. In the next few paragraphs, we discuss more as to why this particular simulation platform was chosen [87].

OS3 or CNI_OS3 implemented simple satellite mobility (such as SatSGP4) without any satellite communication and satisfied all our other requirements. The foundation of this work was Galileo Satellite Communication Simulator (GSCS) [88] (also known as Multi-scale Satellite Simulation Environment (MSSE)). Although GSCS was based on the INET framework of OMNET++ simulation engine, it was Galileo satellite navigation system specific. CNI_OS3 uses a TLE (Two-Line Element) format file for

fetching initial positions of various satellites being used for simulation. Depending upon the specific Navigation System TLE file used, that particular navigation system can be simulated. For example, we can use TLE file of 31 GPS satellites to simulate GPS while we can use a TLE file of 30 Galileo satellites to simulate Galileo Navigation System. It provides an accurate satellite movement simulation with live weather data, high-resolution altitude data, different visualization options, etc.

3.3 Constraints and Assumptions

Various constraints and assumptions made during the development of GPS in UAVSim include the following:

- *Generic computing environment*: The project being non-funded, available medium-end systems were used rather than the high-performance parallel computing systems. Only open source simulation environments could be used because of the same reason.

- *System information and validation data non-availability*: Most of the system information is not available in the public domain and thus gathering various network and communication-related information posed a significant challenge.

- *Distance measurement*: In the GPS implementation, instead of calculating the distance between the satellite and the host using the speed of light and time difference in transmission and reception, this distance is being sent in the packet itself. Reason being the limitation of OMNeT++ of Tx/Rx event timing being exactly the same (accurate up to a nanosecond) to make it appear real-time.

- *Trilateration*: We have approximated the implementation to a 2-D localization (trilateration) instead of 3-D (multilateration). This is due to the limitation posed by UAVSim, CNI_OS3, and the underlying simulation engine OMNeT++.

- *Attacker capability*: Capability of attackers have been assumed to be equal or more than the UAVs in simulations. This enables impact evaluation of more powerful adversaries as well as scenarios in which our systems are compromised.

- *Radio disturbances*: The communication environment of the simulation testbed takes into consideration disturbances caused by random noise, upper layers of atmosphere and communication signals present in the lower layers.

3.4 UAVSim Design

The simulation testbed is developed using the open source network simulator OMNeT++ and one of its independently developed open source modules called INET. Network design and higher level modules are coded in NED, a language specifically designed for OMNeT++ while the lower level functioning is coded using C++ [85]. Although it has an inbuilt GUI and a result analysis module, we developed another GUI to make it more user-friendly. Several attacks have been implemented in the attack library of the testbed. Further, an advanced model of UAV has also been designed, and external models could also be coded in the simulation environment. As mentioned earlier, the interactive GUI of the testbed lets the user change various parameters while advanced users can directly edit the configuration (*.ini) files. The testbed supports mobile wireless communication, UAV component level modeling capability, and detailed network analysis at lower levels of the protocol stack. Further, attacks targeting different layers can also be designed and tested in the testbed. Another important feature of this testbed from the user perspective is its user-friendly design and its ability to work on the generic computing environment. Figure 3-1 shows the high-level architecture of the experimental testbed. We used OMNeT++ version 4.2.2 with version 2.2 of INET.

An OMNeT++ network model is composed of hierarchically nested modules that

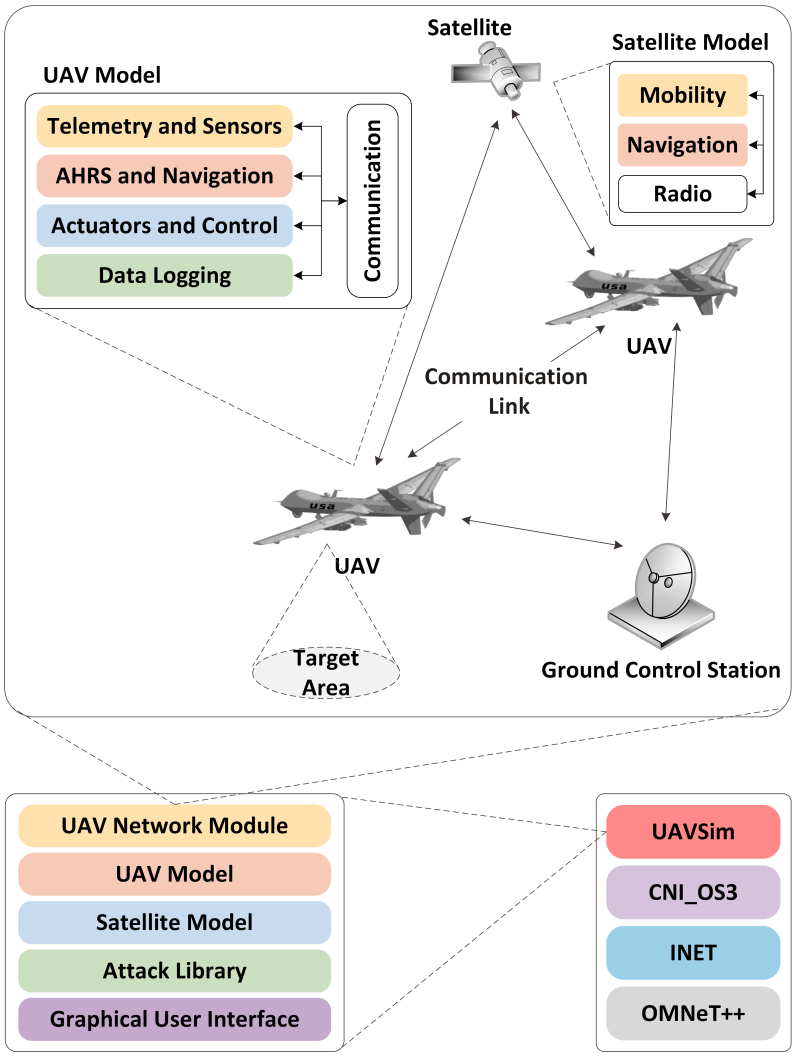

Figure 3-1: UAVSim simulation environment architecture

communicate through a message passing mechanism. Various modules can send messages to their destination along a predefined path directly, or through gates and connections. OMNeT++ has a separate mobility framework that supports node mobility, dynamic connection management, and a wireless channel control [85]. Other than this mobility framework, there is another mobility module called INET which has extensive modules for simulation of various wireless protocols including implementations of various mobility and radio propagation models [89]. We have used OMNeT ++ 4.2.2 and inet 2.0 for our development. The network is defined in NED (Network

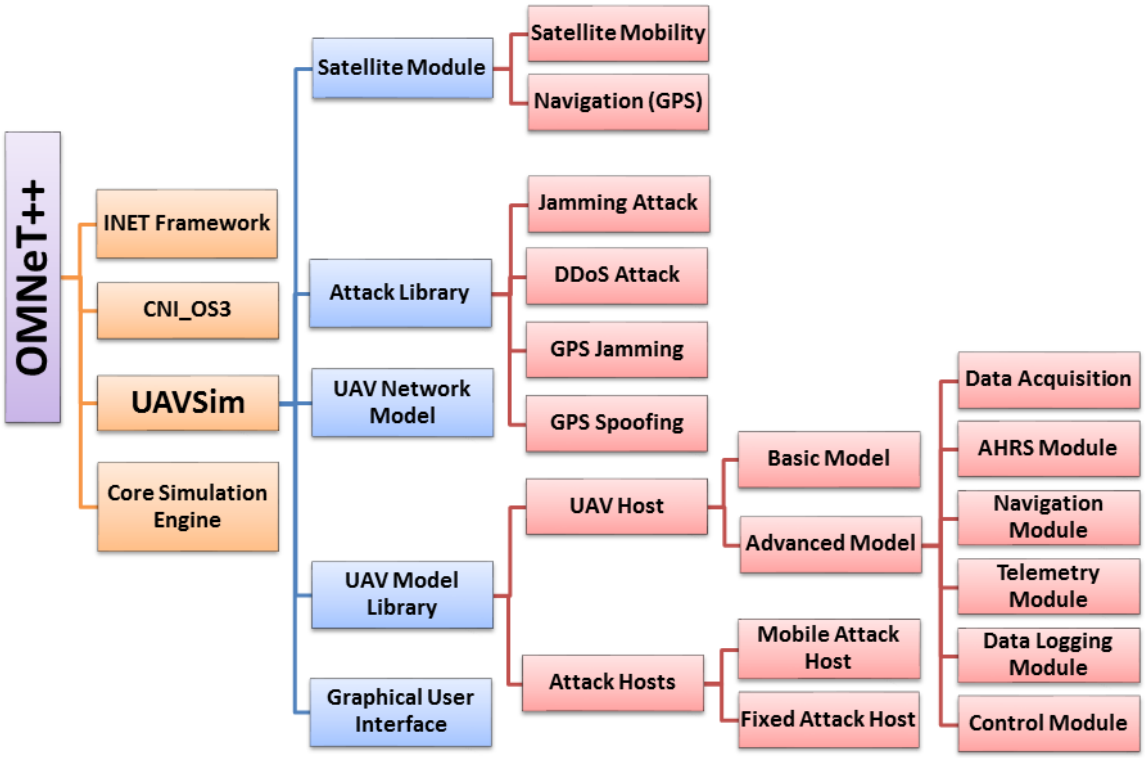

Figure 3-2: UAVSim modules

Description) files using NED language (specifically designed for OMNeT++). This NED file imports various packages required for the simulation of the network. The second important component is a configuration file, usually named as omnetpp.ini, which contains the description of all network parameters including protocols, time to live, duration of simulation, position of nodes mobility model being used, number of hosts, etc. This file is used to initiate the simulation, and it also contains various UAV and attack specifics that are allowed to be modified by the user. In UAVSim, the GUI sets the various parameters and users don't need to edit this file. Also, advanced users can still directly edit this configuration file in UAVSim. Figure 3-2 show various UAVSim modules and how they are placed in the design.

3.4.1 UAV Model Library (UAVModel)

UAV model library contains all UAV model definitions as well as the attack host definition. Since attack hosts can be wireless or fixed, they are defined in this module as sub-categories of the Attack host sub-module. A proper definition of UAVs was also done in this module using the UAV architecture presented in Section 2.1 which closely resembles the actual UAV. It depicts the UAV as a combination of six components: Data acquisition module, AHRS (Altitude and Heading Reference System), NAV (Navigation) System, Control Module, Data Logging Module and the Telemetry Module. This simple UAV architecture captures the basic functions of a UAV. Additionally, it is assumed that the wireless attack host has a similar capability to launch a successful cyber-attack and hence, the same model is used for the wireless attack host too.

There are two UAV models defined - the basic and advanced. As mentioned earlier, the basic model is similar to a black box and is emulated using a single communicating node. None of the components is identified individually in this model. The advanced model, on the other hand, has six modules and is implemented using different C++ files. These modules communicate with each other using message passing mechanisms implemented in C++ and using the NotificationBoard of OMNeT++. Each module is a separate object (in C++) which sends the required messages to other modules (objects). The message passing is according to the 6-module UAV architecture defined earlier, and two types of message signals are passed: data and control signals. Use of the advanced model increases the simulation by 5-6 times, and that is why we used the basic model in most simulations.

While running simulations, the main simulation file imports this library to create UAV hosts during the simulation. Basic properties of a UAV such as speed, mobility, communication protocol, radio propagation model, etc., are defined for the model and

we also let the user customize/define some of them. For advanced users, this module will also allow users to change default values of some variables, which will require changes in the C++ and NED files. The NED files define the architecture of the UAV and the corresponding C++ files define how these components communicate. The C++ files also define various parameters like time lag, time of response, etc. All defenses against various attacks will have to be defined in these C++ code files.

Most basic attacks can be simulated using the basic UAV model as it defines the core components and communication channels. The advanced UAV model also allows communication between the UAV components and allows changes in parameters related to each one of them. A lot of research in this area involves development, implementation and testing of new protocols and new defensive techniques. This also serves as one of the motivations for developing this testbed specifically for UAVs. Implementation of these new protocols or techniques can be achieved by writing modules in C++ for the protocol and defensive technique.

3.4.2 UAV Network Module (UAVNet)

This module is used to define the various network parameters of the UAV Network, which are specific and unique to the UAV Network. It is known that the protocols used for MANETs are similar to the ones being used in UAVs, but the transmission range, bandwidth and power consumption are quite high. These parameters and other network specifics like the number of radio channel, fixed host properties, protocols being used, maximum noise allowed, maximum transmission power, etc. are defined in this module. Several parameters defined in this module use base packages from OMNeT++ directly, and others related to mobility and radio communication are imported from inet 2.0.

3.4.3 Attack Library (AttackLib)

The Attack module contains all the attack libraries. Apart from using a standard UAV model, selecting appropriate and conventional attacks is one of the most important aspects of ensuring the proper working of the testbed. It can also help the designers of any system prioritize which aspects to address first and accordingly improve the system. Based on the threat model proposed in [15] we selected two most important kinds of attacks for UAV systems - DDoS and Jamming. As mentioned in section 2.5.3, these pose the highest threat and are the most damaging in terms of compromising Availability of the UAV system. Based on the classification of various threats and possible attacks on UAVS, we implemented four major attacks in the attack library of UAVSim, namely, Continuous Jamming, Single Target DDoS, GPS Jamming and GPS Spoofing. GPS Jamming and Spoofing are the latest addition to the Attack Library of UAVSim while other two have been demonstrated in one of our publications [55].

3.4.4 Graphical User Interface (GUI)

GUI is one of the most important sub-modules of UAVSim. This model was specifically developed to allow users to change parameters and run the simulation for specific values. The GUI makes the testbed easier to use and reduces the technical expertise required to understand the underlying architecture and module arrangement. Users can simply set the values they desire, and other parameters are taken as default values to run the simulation. While the simulation is running, it shows the real-time network behavior, which makes it easy to visualize. The attack hosts and the UAV hosts are also represented using different icons that help the user distinguish between them. There are circles around each of the hosts that represent its transmission range. As a future work, another GUI for advanced users can be developed. Currently, these

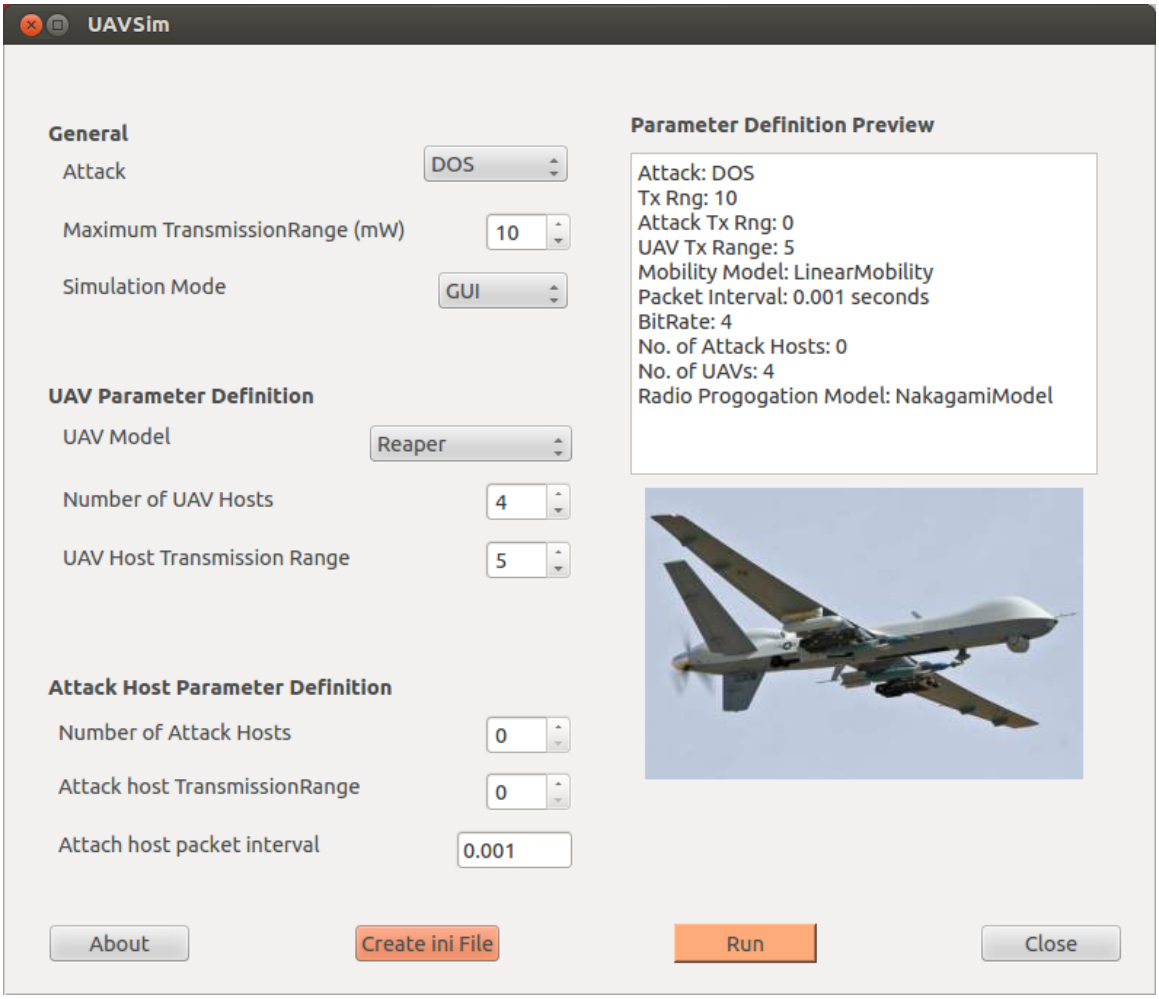

Figure 3-3: User-friendly UAVSim GUI for basic users

other advanced options need to be changed in the configuration file manually which requires more technical expertise. Figure 3-3 shows the basic user GUI designed for UAVSim.

3.4.5 Satellite Module (SatelliteModel)

Satellite model library has the standard satellite model that inherits its basic features from the satellite model defined in CNI_OS3. This module has all the features related to the implementation of GNSS. Each satellite uses a Two Line Element

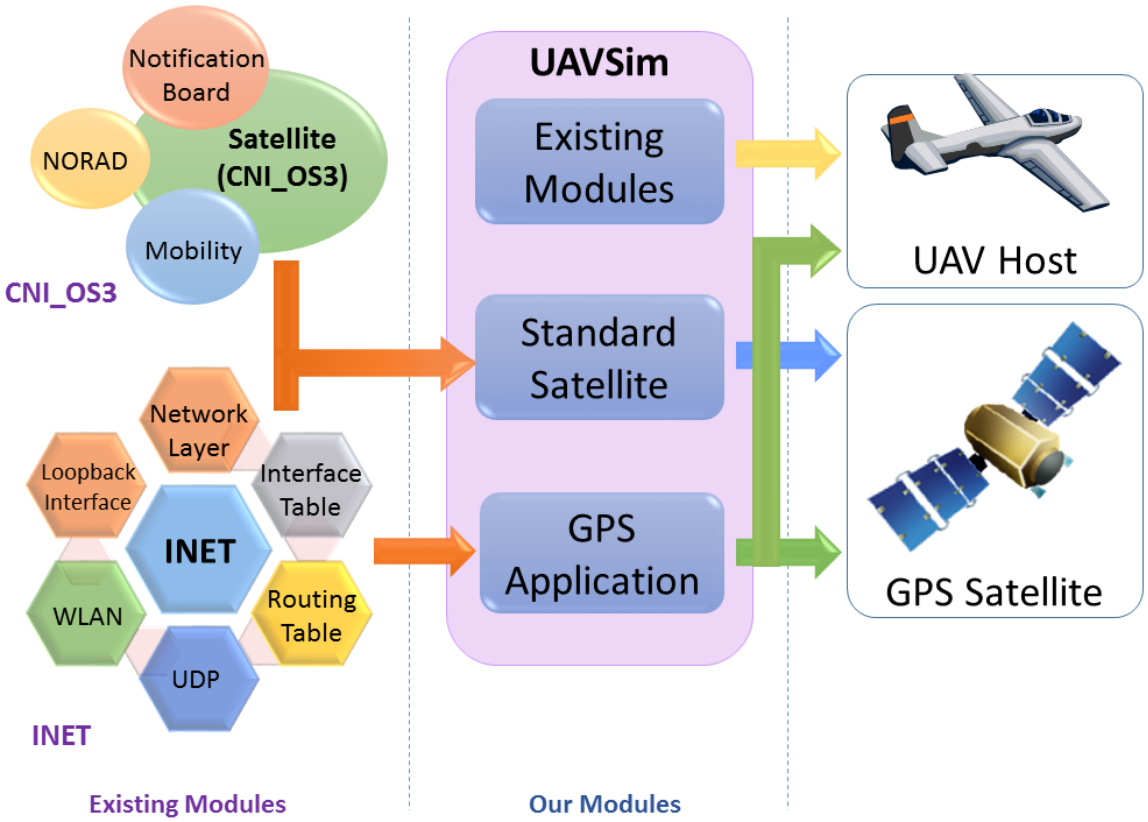

Figure 3-4: Satellite and navigation implementation in UAVSim

(TLE) data, which is a standard format dataset defined by NORAD (North American Aerospace Defense Command), which describes the orbits of all satellites. It should be noted that other than the basic satellite model, CNI_OS3 doesn't provide any communication or navigation-related capability. The GPS functionality has been added to the satellite model through the development of a broadcasting application that sends position information to the receivers through radio signals of the L1 frequency range as per the standard GPS implementation. Figure 3-4 represents the overall design of the Satellite module and navigation. In the next few paragraphs, we briefly discuss each of these components, including existing mobility component.

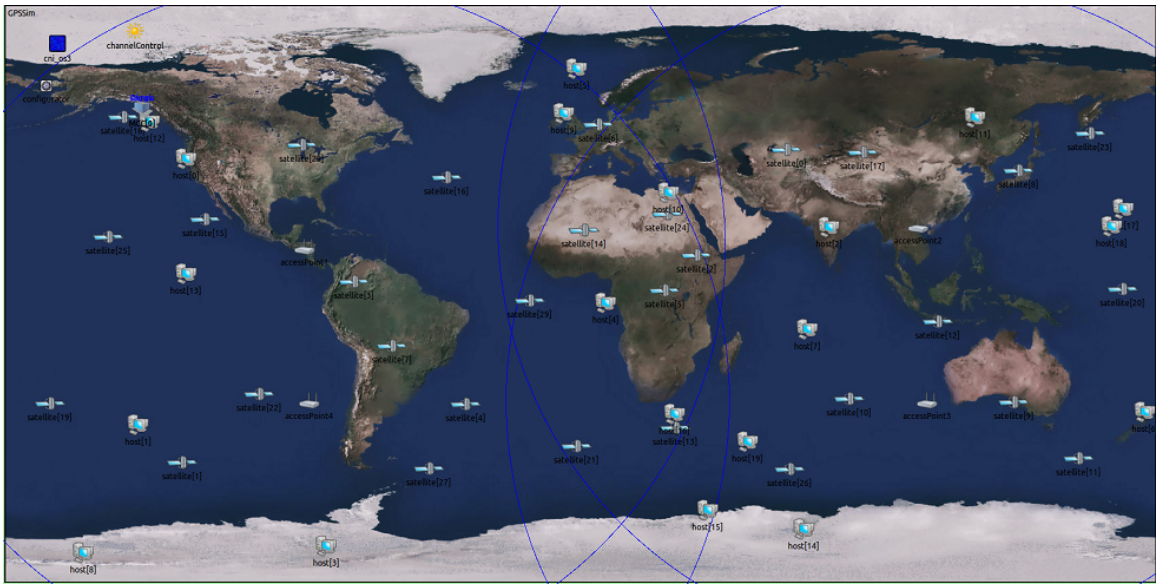

Figure 3-5: World map used for GPS simulations in UAVSim

3.4.5.1 Satellite Mobility

Mobility is the another aspect of the satellite due to which we chose the open source satellite simulator (CNI_OS3). Two types of mobility, SatSGP4, and FishEye-SatSGP4 have already been implemented in this simulator. While FishEyeSatSGP4 is used for a fish-eye view simulation, SatSGP4 is used for regular simulation. SatSGP4 is the standard satellite mobility protocol, and it defines the movement of all terrestrial satellites. Various parameters related to the mobility are fetched from the TLE (Two Line Element) data file of each satellite system. The current TLE file being used is for GPS constellation and contains TLE data for 31 satellites.

Velocity in Simulation: The map pixels have been considered as the position (coordinates) of the satellites and hosts. The scaling of the earth to the map has been done as follows. The radius of the earth is 6371 km, therefore, the circumference of the earth will be

$$2\pi r = 2 * 3.14 * 6371\ km$$

If we imagine opening the globe vertically as a 2-D map as shown in Figure 3-5, the circumference will give the width of the map while the height of the map will be half the circumference. The earth map used in our simulations have dimensions as

$$1080 * 2160 \ pixels$$

Therefore, each pixel on the map corresponds to

$$\frac{2 * 3.14 * 6371 * 10^3}{2160} = 18.5 \ km$$

Therefore,

$$1m = \frac{1}{(18.5 * 10^3)} \ pixel$$

Different UAV models available today have different speeds depending upon their design and usage. The Arcangel-1 has a speed of 150 km/h while Airforce mission UAVs, like Patroller, can fly at a maximum speed of 240 km/h. Taking an above average speed of 250 km/h (69.5 m/s) of a UAV, the speed of the satellite on the simulation map will be

$$69.5/(18.5 * 10^3) = 0.0037 \ pixels \ per \ second$$

This is the speed we use for all simulations. Thus, a speed of 0.0075 pixels per second on the map would correspond to a speed of 500 km/h in the real world.

3.4.5.2 Satellite Network Module (GPSSimulation)

Similar to UAV network module, this module defines the network stack of satellites. It defines the communication protocol, transmission power, access points, etc. This module uses several basic packages from INET for satellite communication while

the satellite mobility packages are used from OS3.

In this component, we defined the communication network stack for satellites. We modified the existing satellite model to make it communication capable. Various libraries from INET and OMNeT++ including communication protocol were used to implement this module. Different variables that can be modified by the user were also defined in this model, such as transmission power, differential GPS stations (access points), data burst duration, the gap between data bursts, etc.

3.4.5.3 Navigation Module (GPSApp)

The main component of a navigation module is its GPS unit that helps it to know its actual position. It also helps the UAV during different preset navigation modes, such as position hold, return-to-home, autonomous flight and collision avoidance [90]. Depending upon the area in which the UAV is deployed, a GPS can be used to link data to its spatial position. This method is called geo-referencing [91]. It is due to their navigation capability that UAVs have found their application in various fields and operations. The GPS signals are typically very weak, sometimes less than 100W and are transmitted over a range of $20 - 25,000$ kilometers. This low power makes them fall below the noise floor spectrum when they reach the earth's surface [92]. These signals are vulnerable to failure, disruption, and unintentional or deliberate interference. Clearly, navigation is one of the most critical modules of a remotely controlled UAV. The increased dependency of UAVs on GPS signals for localization, navigation and time-synchronization has made it a focus area for adversaries and thus led to the discovery of its vulnerabilities to attacks like Spoofing and Jamming. Simulations related to UAV operations involving GPS and related navigational aspects are quite important for correct simulations.

The navigation module has the receiver end GPS application that enables the hosts to receive satellite navigation signals carrying its position information. The

information from four or more satellites is required to calculate the position of the UAV using multilateration in the 3-D space. In our implementation, we have used trilateration for localization. Therefore, a minimum of three satellites are required. Navigation functionality has been added to the satellite model through the development of a connectionless, no-reply broadcast application. This application sends positions information to the receivers through radio signals of the $L1$ frequency range as per the standard GPS implementation. The GPS receiver application has also been implemented separately in hosts that enables them to receive satellite navigation signals carrying position information.

Packet Structure and Functioning: The space segment constitutes of GPS satellites while the user segment in our simulations would be the UAV hosts. The GPS application in the satellite creates and broadcasts the packet frames. On the UAV host, the GPS application receives the packets from various satellites and processes this received data for localization. The communication between UAVs and satellites or any GPS receiver is unidirectional, from satellites to the host. Satellites broadcast signals without waiting for any acknowledgment, and that is why a connectionless broadcasting protocol have been used. The packets contain the satellite index, X coordinates and Y coordinates of the satellite sending the packet, and the distance from that host. For the first time when a UAV receives the packet, it locks on a fixed number of satellites. After obtaining a lock on three different satellites (in 2-D implementation), the UAV calculates its position. Further, it calculates its position whenever it receives three more packets from these locked satellites.

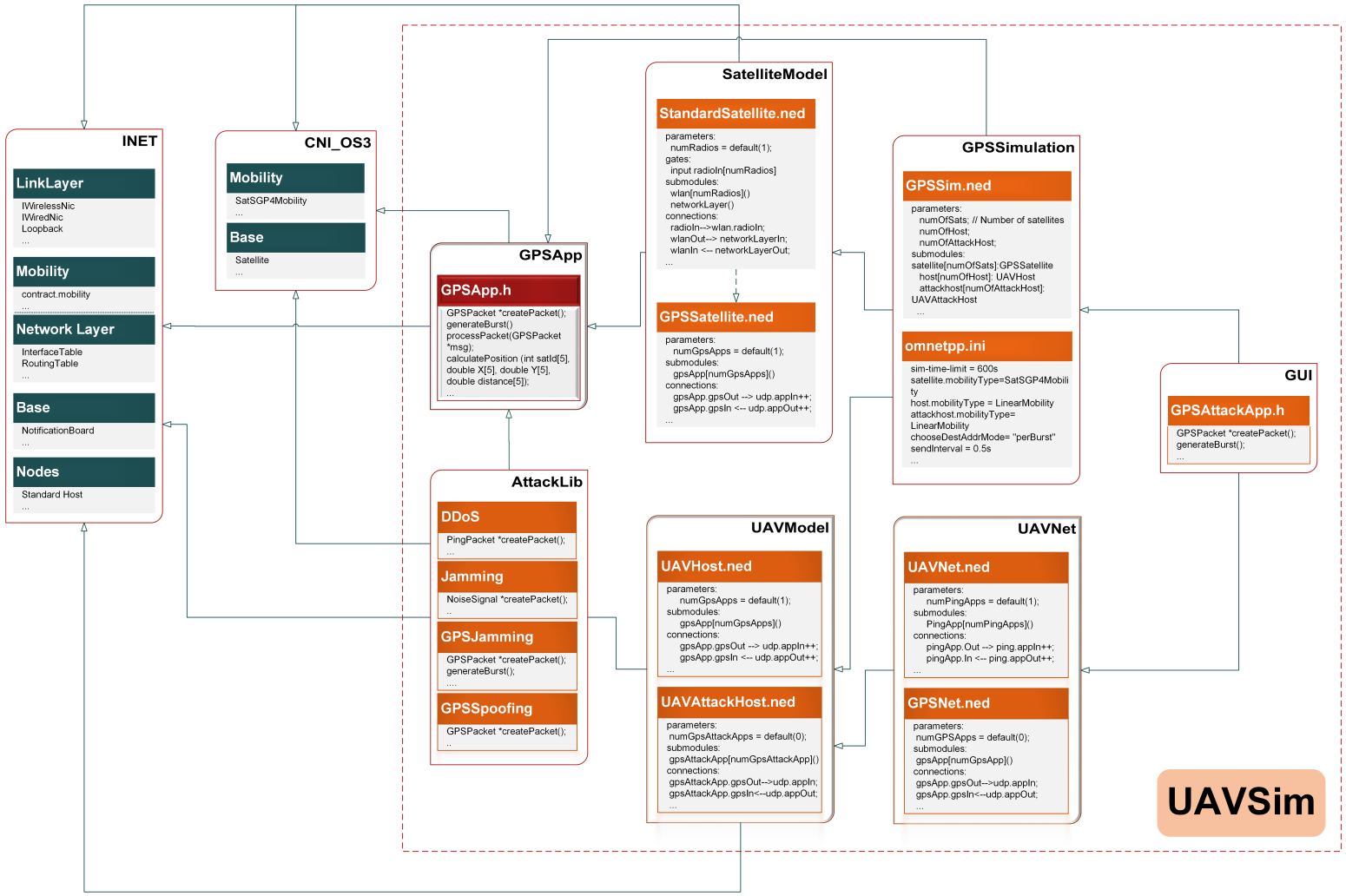

Figure 3-6: Class diagram of UAVSim

3.4.6 Additional Features

As mentioned in the previous section, the primary aim of designing and developing a software based simulation testbed was to evaluate the behavior of a single UAV in case of attacks. Due to the use of large number of UAVs nowadays, it becomes essential to assess the performance of these UAVs in a swarm setting. Developing such a testbed becomes a necessity when the authorities are planning to integrate UAVs in the National Airspace (NAS). Keeping all these requirements in mind, we initially worked on UAV component level modeling, individual simulation, attack classification, and attack modeling [39]. Later, we developed a software simulation testbed called UAVSim for simulations of all sizes of UAV networks [43]. Clearly, this testbed is not comprehensive and needs the addition of various UAV models, attacks and their detection or mitigation measures.

One of the most important feature and primary focus of UAVSim is the security simulation for UAVNet. Several attacks have been implemented in the attack library of the testbed. Further, basic and advanced models of UAV have also been designed as well as the facility of using external models is provided. These external models are usually XML based and developed by other researchers. As mentioned earlier, the interactive GUI of UAVSim lets user vary various parameters while advanced users can directly manipulate the configuration files. Most performance tests were performed for security simulations and are reported in chapter 5. Attacks targeting distinct layers of the protocol stack can be designed, launched and tested in UAVSim. From the user perspective, one of the most important features of UAVSim is its user-friendly design and its ability to work on the generic computing environment. Figure 3-7 summarizes these important features and modules of UAVSim.

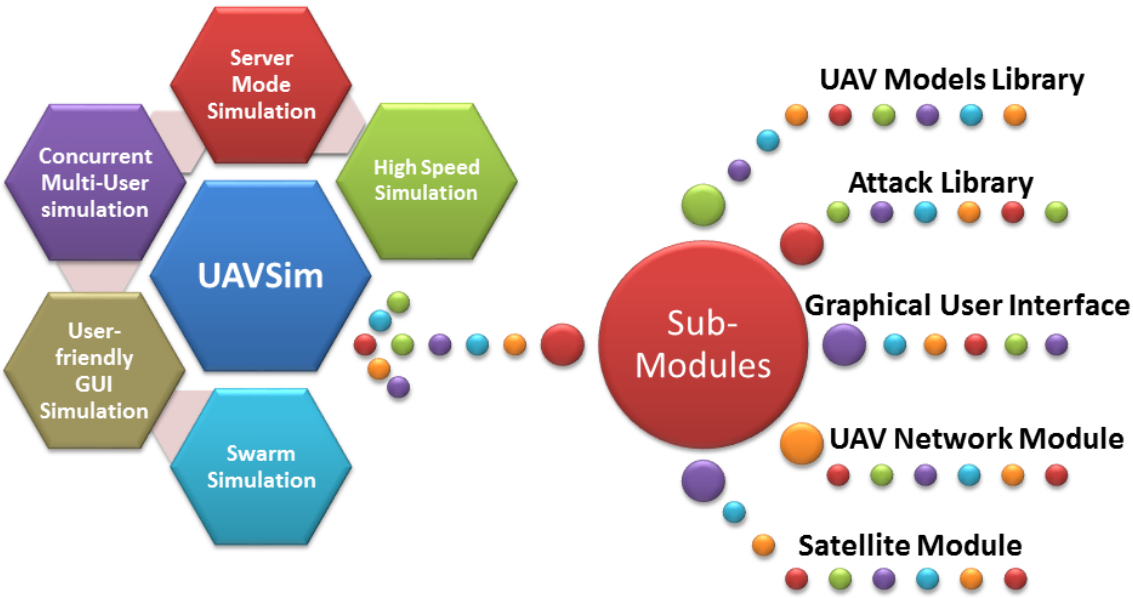

Figure 3-7: Modes of operations and various components of UAVSim

3.4.6.1 User-friendly GUI Simulation

UAVSim supports command line as well as GUI-based simulations. We have also developed a custom basic GUI for UAVSim, which allows basic users to set different parameters, according to their requirements. Clearly, users won't get a lot of independence in this GUI. The advanced users, on the other hand, can edit all the parameters by directly editing the *omnetpp.ini* configuration file of the particular simulation project. Use of the GUI might cost some computational resource, and that is why we have evaluated testbed performance using it as one of the performance parameters.

3.4.6.2 Server Mode Simulation

To enhance the runtime performance of the testbed, a high-performance computer can be used to run UAVSim. Remote connection details (such as IP, username, password, etc.) for a high-performance computer or a server can be set in the GUI by

the administrator for initial use. It is important to know that the core testbed files should be installed on the server prior to this connection setup, and ssh should also be enabled on the high-performance computer. These prerequisites ensure seamless communication and simulation execution.

3.4.6.3 High Speed (No-GUI) Simulation

We already discussed that we developed a user-friendly GUI for UAVSim. The aim of providing a non-GUI option with a GUI was to enhance the simulation performance of UAVSim. There is another option of command mode express execution that prints the minimum required simulation statistics to let the user know that the simulation is running. Using this option, the simulation can execute at the maximum speed and thus, gives the best performance. This mode is primarily available for server mode simulation because the communication with the server might slow down execution. Nevertheless, this mode can be used on the desktop mode as well as server mode.

3.4.6.4 Concurrent Multi-User Simulation

A multi-user option has also been provided in the testbed that allows multiple users to run their simulations concurrently on their PCs. This option is based on the server-mode simulation. If the testbed needs to be used for high-speed simulation in an academic class or lab setup by several users simultaneously, a non-GUI command-line option is available. One of the prerequisites to use this option is the connection-oriented access availability on the server to all the users accessing it. This network access is mandatory to enable independent simulation of each user. The core simulation modules need to be installed beforehand on the server while users connect to the server remotely through ssh. Once configured with the connection details to the server, UAVSim automatically connects to the server and displays results in a console window.

3.4.6.5 Swarm Simulation

UAVSim also supports UAV swarm simulation just like any other network simulators. This particular feature allows users to test the UAVNet behavior when large numbers of UAVs are deployed for an application. This application can be commercial, civil or military in nature. Typically, the swarm of any vehicle is used in wide area sensing applications or application where a large number of sensors are required. UAVSim performance for this swarm simulation feature has also been evaluated for UAVS using single and multiple frequencies.

3.5 Chapter Summary

This chapter primarily details the design and implementation of the proposed simulation testbed UAVSim. Initially, requirements for such a testbed along with candidate evaluations and various limitations encountered were discussed. Subsequently, the five major modules of UAVSim, UAVModel, UAVNet, AttackLib, GUI and SatelliteModel, were discussed in detail. Finally, few additional features were also described, which would enhance the usability and capability of this testbed.

Chapter 4

Attack implementation and Analysis

This chapter consists of the core theme of this dissertation. Once the simulation testbed was developed, the obvious next step was: security simulation. To accomplish this task, we primarily implemented two classes of attacks: Availability and Integrity compromising attacks. As stated in section 2.3, the design of confidentiality compromising attacks requires an enormous amount of intelligence gathering by the adversary in terms of security mechanisms that could prevent the attacks. Therefore, such attacks are very sparse in this domain. Simultaneously, it has been observed that availability attacks are most common and pose a greater threat for UAV Networks. Further, a detailed risk analysis also indicated the impact severity of availability attacks in UAV systems. These reasons served as the primary motivation behind choosing the attack categories mentioned above.

Two specific attacks were chosen in these two classes. For Availability, we chose DDoS and Jamming while GPS Spoofing and GPS Jamming attacks were chosen in the second attack class. It should be noted that the GPS Jamming attack also falls in the Availability attack class. In essence, it is a type of selective jamming which attempts to specifically jam GPS signals. It has been considered as an integrity

Table 4.1: Default values of some simulation parameters

Parameter	Value
Simulation Time Limit	300 seconds
Radio propagation model	Nakagami model [93]
Mobility Type	Linear mobility
Packet interval for UAVs	0.05 seconds [94]
Packet interval for attack hosts	0.0001 seconds
Number of UAV hosts	10
Number of attack hosts	30
UAV transmission power	5 Watts
Attack host transmission power	10 Watts

attack because it introduces the vulnerability of false GPS signal acceptance in the UAV. The following sections discuss the implementation and detailed result analysis of these four attacks.

4.1 Availability Attacks

This section discusses detailed implementation as well as various results for both availability attacks: DDoS and Jamming. Using the testbed, we analyzed the effect of changing various parameters that might affect the overall UAV System in a multi-UAV environment. Average loss and average round trip time were calculated by averaging them for all the hosts in the network. These two quantities represent typical network performance and indicate the reliability and availability of time-sensitive systems. Some of the default parameters used for these simulations are defined in Table 4.1.

4.1.1 DDoS

The DDoS (Distributed DoS) attack aims at network congestion or overflow in the buffer memory of the network interface card, to make the host appear unavailable or offline to all other hosts in the network. It is a well-known attack where services

or access to resources is compromised by an increase in the round trip time than the time to live. This increase results in an high packet drop rate. In our attack implementation and simulations, we have used a hybrid approach for launching this attack by transmitting a large number of packets from various attack hosts such that the buffer memory of the wireless network interface card also overflows.

4.1.1.1 Implementation

Th DDoS attack has been implemented in UAVSim using a large number of attack hosts. This number can be set by the user based on the total number of UAV hosts in the network such that the attack works. Several cases are possible, including, testing an attack and testing a defensive technique. It has been experimentally proved that a single attack host is capable of launching a successful DoS (but not DDoS) attack using ping packets because of its small payload size [95]. All attack hosts behave similar to regular UAV hosts and are assigned the IP addresses of the same range to make them indistinguishable from other trustworthy UAVs. During the simulation analysis, we have varied this number to check the success rate of attack in different scenarios. All attack hosts transmit packets to a single UAV host to make it unavailable and thus, result in a successful attack. By default, the number of attack hosts is 30, which was calibrated using several simulation experiments. Approximate time taken to successfully launch this attack is only a few seconds for all simulations and packet loss for the attacked host reaches 99.9% in less than 2 seconds.

4.1.1.2 Simulation results

For DDoS attack, we discuss effects of three basic parameters on attack performance - the impact of mobility, increasing number of UAV hosts, and transmission range variation of both hosts and attack hosts. For DDoS attack, we assume that the attacker has already gained information about addresses being used in the system

and thus, can launch a DDoS attack using IP spoofing. Throughout this section, irrespective of how many hosts are present, please note that host1 is the host that is being attacked, and host2 is the host not being attacked. The performance of hosts2 is being monitored for the sake of comparison as all other hosts communicate with these two hosts.

4.1.1.2.1 Effect of Mobility

Movement of any host in a wireless network plays an important role in the communication due to the reason that devices usually have a limited range and they won't be able to communicate with each other once out of range. Therefore, for the first analysis, we chose four communicating hosts such that all of them have two wireless network interface cards. These hosts (except host1) use the first interface to communicate with host1 while the second wireless interface card is utilized by all hosts (except host2) to communicate with host2. The attack is being attempted at host1, and the corresponding results are presented here.

Figure 4-1 and figure 4-2 show the average loss of the host1 with increasing number of attack hosts for different mobility models. The average loss is defined as the mean of packet loss of all the hosts in the network that are trying to connect to the host under attack, i.e., host1. As shown in figure 4-1, simulation were ran using different mobility models, and it was found that in all cases, Mass mobility and Linear mobility models have the least average loss in the network. Therefore, we tried to analyze these two models in more detail using more attack hosts. As shown in figure 4-2, these two models also reach close to 100% average packet loss when the number of attack hosts is increased to 30.

Figure 4-3 shows the average round trip time (RTT) for the network as the number of attack hosts is increased. It should be noted that the average RTT increases from 1.69 seconds to over 200 seconds for as less as 16 attack hosts. As the number of attack

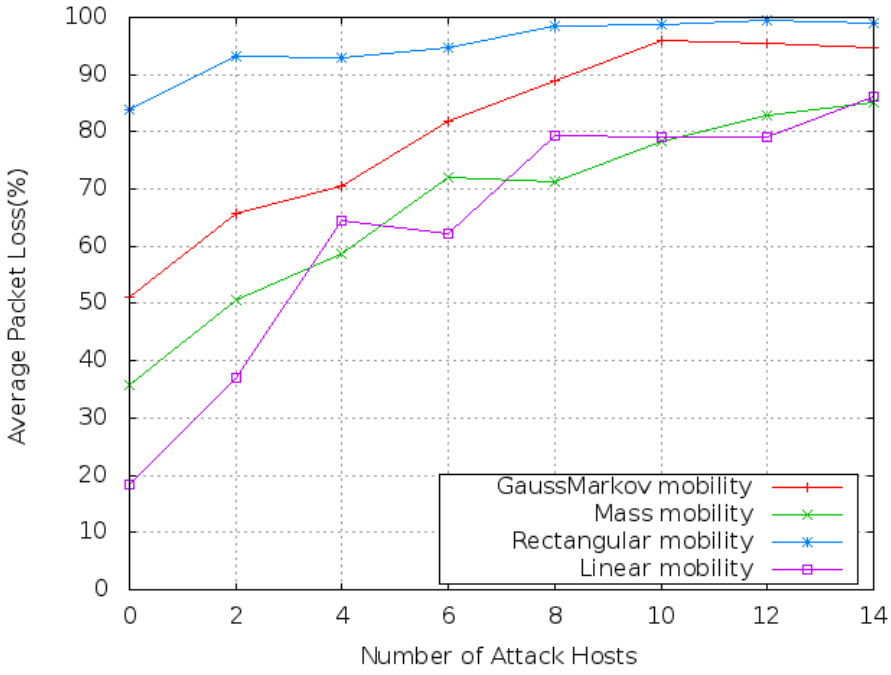

Figure 4-1: Average packet loss for DDoS attack using four mobility models

Figure 4-2: Average packet loss for DDoS attack using linear and mass mobility models

Figure 4-3: Average round trip time (RTT) for DDoS attack using linear and mass mobility models

hosts increases to 30, the average RTT increases to around 500 seconds, i.e., more than 8 minutes. This value exceeds any possible limit of RTT and thus, indicates that there will be massive delays even if RTT limit is increased.

4.1.1.2.2 Effect of Increased Number of UAV Hosts

In this case, we have fixed the number of attack hosts as 20 and attempted to analyze the impact of increasing number of UAV hosts in the network, on other communicating hosts that are not under attack. Of course, one would think that there should be no effect on other hosts since the attack is being targeted on another host. Unlike the traditional wired network, there should be no network congestion. Surprisingly, we see that the presence of attack hosts in the network has a huge effect on other communicating hosts. Concurrently, we compare it with losses in both these hosts when there were no attack hosts.

We keep increasing the number of regular hosts to see the impact on hosts com-

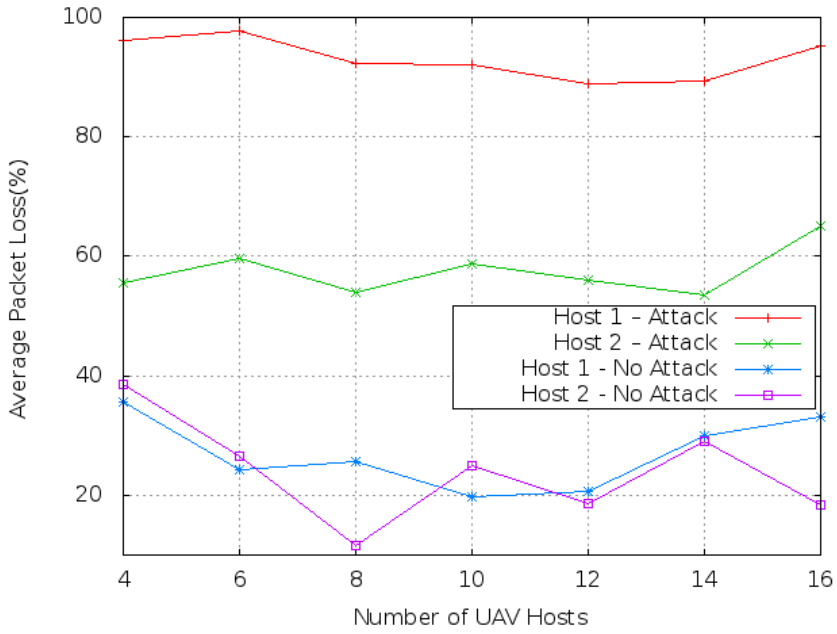

Figure 4-4: Average packet loss for two hosts during attack and under normal conditions following mass mobility

municating to host1. It has been shown in previous mobility analysis that in the absence of any attack host, the linear mobility model performs the best; therefore, we used the same mobility model for this study. Although rectangle mobility is quite close to the actual UAV movement in real world UAV applications, using it would not give us a clear idea of the impact because of existing massive losses.

Figure 4-4 shows the analysis result for Mass mobility model when the number of regular UAV hosts was varied from 4 to 16 in increments of 2. As shown in figure 4-5, we plotted the loss for the two hosts when they were using linear mobility model in the presence as well as the absence of attack hosts. The number of attack hosts, in this case, was also taken as 20 while the number of regular UAV hosts was varied from 4 to 16. As an additional evaluation parameter, we also calculated the RTT for each of these simulations. Figure 4-6 shows the average RTT for these mobility models with increasing number of UAV hosts.

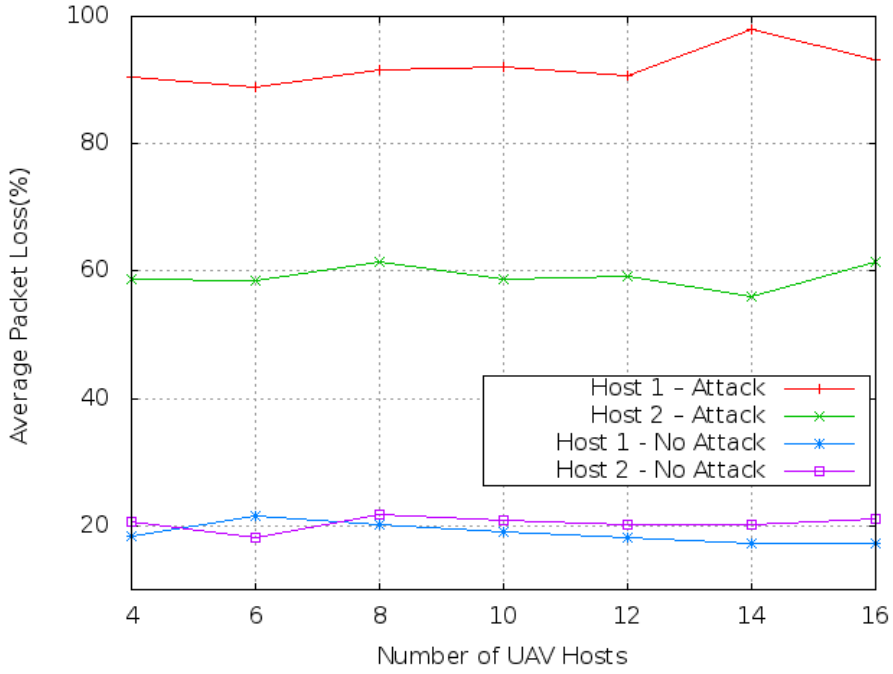

Figure 4-5: Average packet loss for two hosts during attack and under normal conditions following linear mobility

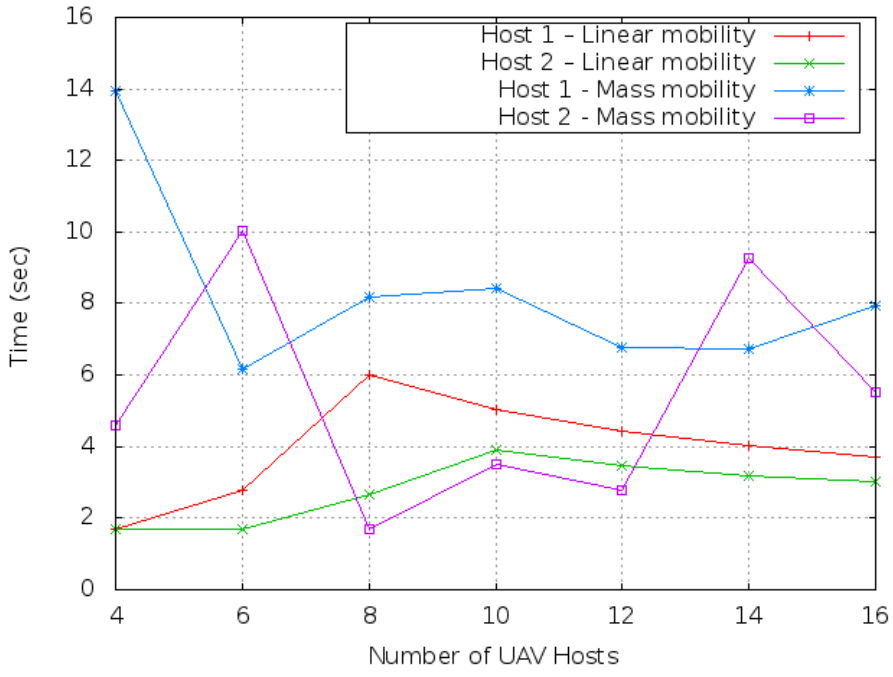

Figure 4-6: Average RTT for mass and linear mobility for two hosts during attack and under normal conditions

4.1.1.2.3 Effect of Transmission Range Variation

In this analysis, we tried to evaluate the effect of transmission power variation in the UAVNet. Due to the wireless nature of links and their dependency on the transmission power, it is important to know the effect of transmission power variation on communication. Initially, we attempted to evaluate the effect of increasing transmission power of UAV hosts from 1W to 10W, where 10W is maximum. As it can be seen from figure 4-7, that the loss in the host being attacked reduces from about 100% to 75% while the attack host transmission power stayed fixed at 5W. This indicates that even if the attack host range is 50% of the regular UAV host, the loss can't be avoided. The average loss for regular hosts that are not being attacked reduces linearly though.

Figure 4-8 shows the results for the case when the transmission power of attack hosts is increased from 0 to 10W while the regular UAV host transmission power is fixed at 5W. The figure clearly shows that the average packet loss increases rapidly when the transmission power of attack hosts is increased up to the transmission power of UAV hosts and becomes constant afterward. Another important trend to note is the loss in hosts, which are not directly attacked, remains almost constant in the range of 50-60%. Even though this loss increases from 18% (when there was no attack), increased transmission power does not have much effect on it once the DDoS attack is successfully launched.

4.1.2 Jamming

Jamming is another major attack that belongs to the category of attacks affecting system availability. It involves the transmission of random noise signals in the mission area to make all communications difficult. The random signals usually cover a broad frequency band and thus, renders all communicating hosts in the area unable to com-

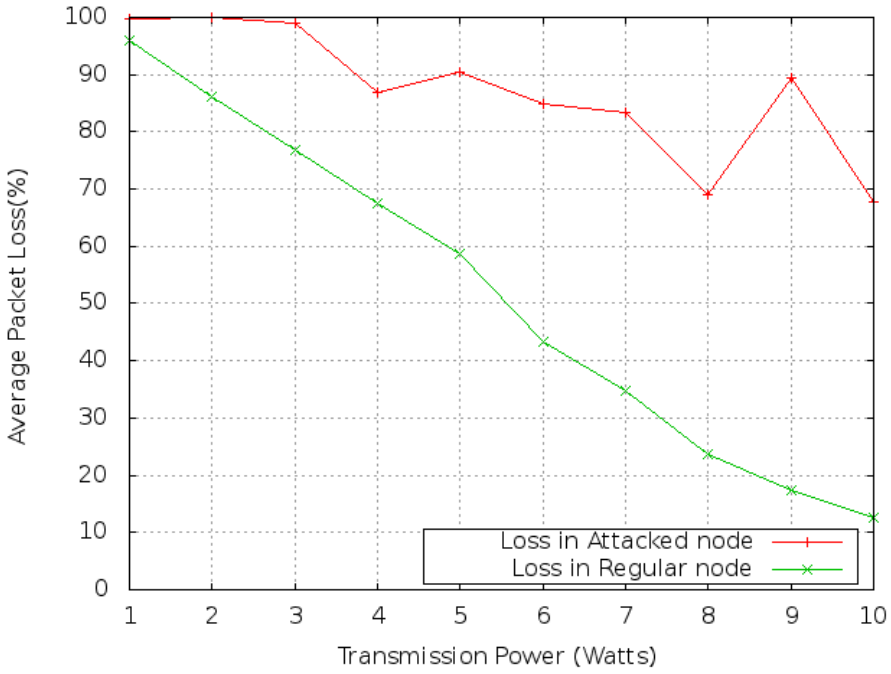

Figure 4-7: Average packet loss variation with increasing transmission range of UAV hosts while attack host range is fixed at 50%

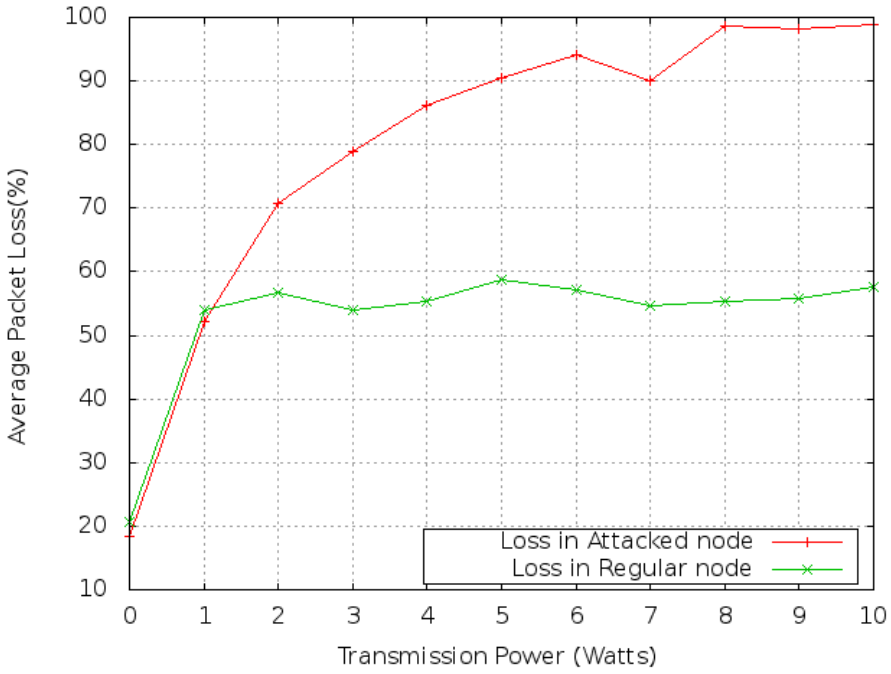

Figure 4-8: Average packet loss variation with increasing transmission range of attack hosts while UAV host range is fixed at 50%

municate. This attack can't be handled by most of the today's wireless devices and is relatively easier to launch. The jamming attack is implemented in UAVSim using another custom designed program, which enables sending random packets to all UAV hosts in a round robin manner. Since none of the network simulators we evaluated had the capability of launching a Jamming attack, we used this method to implement this attack. The working of this module was verified through experimentation. All the attack hosts operate at different frequencies and try to block a wide frequency range in the region to result in a Jamming attack.

All wireless communication in an area can be blocked using Jamming. This attack results in loss of communication through corruption or loss of packets. The noise signal usually traverses over all the possible frequencies being used for communication and prevents transmission and reception at any frequency. If the attack host has an advanced transmitter, a noise signal can be generated that will be powerful enough to overwhelm the targeted signals and interrupt communications. The most common types of signal used in a jamming attack are random noise and pulse [96]. Jamming equipment is readily available in various online shopping websites like Amazon and eBay. Jamming equipment can also be mounted on towers to target networks at a remote location.

4.1.2.1 Implementation

Jamming is implemented in UAVSim by using several attack hosts that send noise signals to all the hosts in a round-robin manner over different frequencies. The number of attack hosts can be changed before running the simulation. Transmitting these random noise signals to all UAV hosts within the range, will launch a jamming attack as aimed. Several researchers have developed techniques to mitigate the effects of jamming attacks. Most of these methods employ concepts like frequency hopping and spread spectrum communication [76]. This is also one of the reasons why we

implemented jamming for a range of frequency in UAVSim. Post-2010, several researchers have proposed anti-jamming encoding, encryption, etc., however, we have not addressed those techniques in our jamming attack implementation due to lack of their use in the real world. To successfully launch this attack, it took 4 seconds longer than the DDoS attack while packet loss for all hosts reached above 90% for all hosts.

4.1.2.2 Simulation Results

We have six UAV hosts where three hosts use 5 GHz band and the others use the 10 GHz band. Two hosts in each group are communicating with the third UAV host. For jamming attacks, we first evaluated the effect of increasing transmission power of attack hosts on the overall UAVS, without varying the number of attack hosts. Further, we evaluated the effect of varying number of attack hosts on the UAVS while the transmission power of all hosts were fixed. All hosts and attack hosts use the linear mobility model as it was earlier proved to work best during attacks.

4.1.2.2.1 Effect of Transmission Power

In this evaluation, the transmission power of attack hosts was varied from 0 to 10W while the regular UAV host transmission power was fixed at 5W. We recorded average packet loss as well as average RTT for all the transmitting hosts. Figure 4-9 shows the trend of average packet loss, and it should be noted that the average packet loss keeps increasing almost linearly with increasing transmission power of attack hosts. On the other hand, the trend of RTT for these simulations is also interesting. As evident from figure 4-10, RTT stays well below the maximum allowed limit of TTL (in IP networks) until the transmission power of attack hosts is less than that of regular UAV host, i.e., 5W. Once the transmission power of attack hosts exceed this value, the RTT increases rapidly to the range of $10 - 25$ minutes. Another important

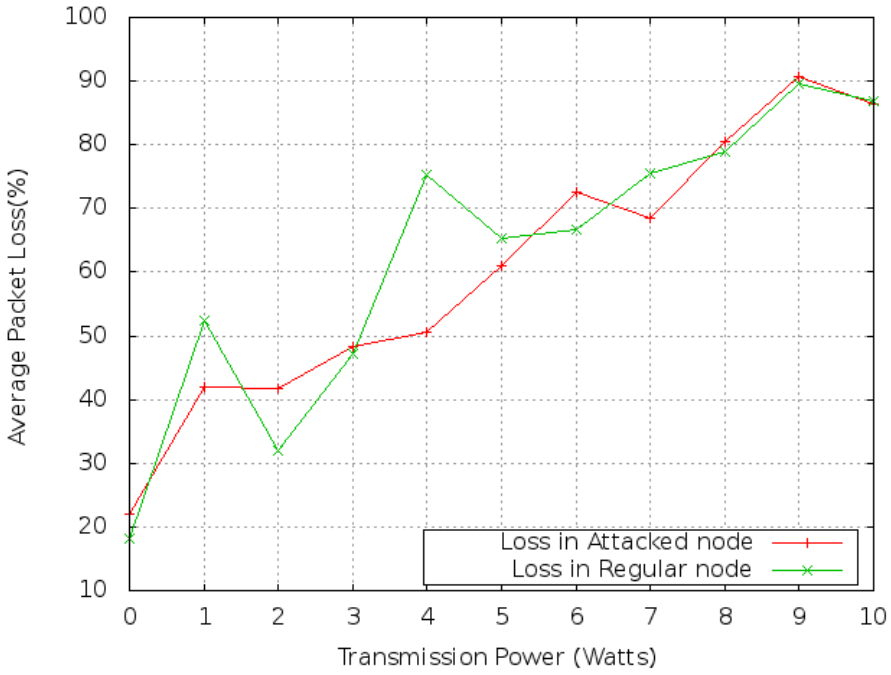

Figure 4-9: Average packet loss variation with increasing transmission power of attack hosts while UAV host power is fixed at 50% (or 5W)

observation is an almost equal impact on both the attacked and non-attacked UAV hosts.

4.1.2.2.2 Effect of Increasing Number of Attack Hosts

Effects of increasing number of attack hosts were evaluated while keeping individual transmission power constant for each type of host in simulations. We consider three cases. Case I, where the transmission power of the attack host is 10W, and that of regular UAV host is 5W. Case II, where the transmission power for the attack host and regular UAV host is 5W and 10W, respectively. Case III, where transmission power of both the attack host and the UAV is 2W. Case I results are shown in figures 4-11 and 4-12. It is clear from figure 4-11 that average packet loss rapidly increases as the number of attack hosts are increased and crosses the 80% mark.

While figure 4-12 indicates that the RTT increases exponentially with the number

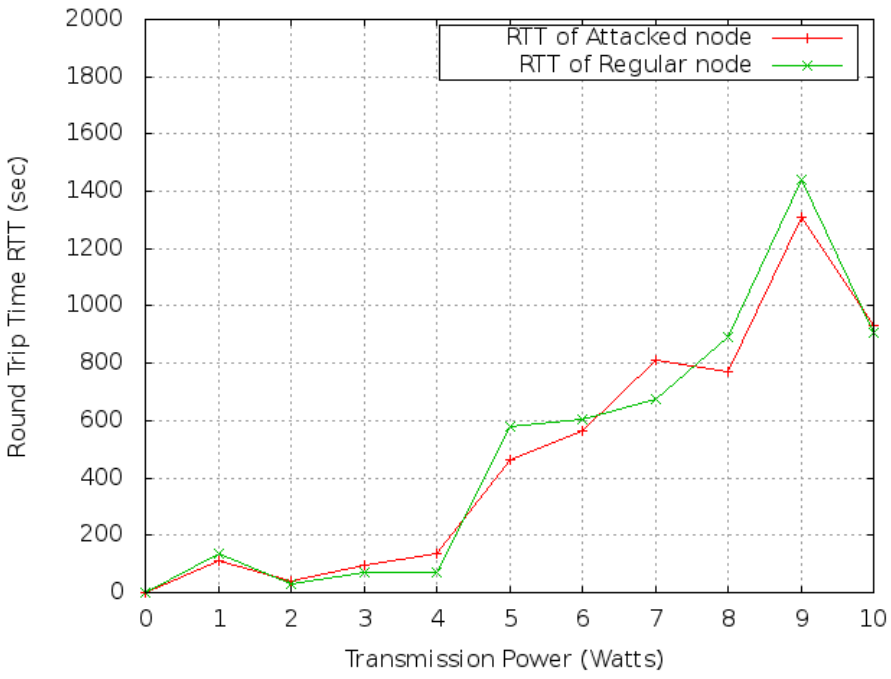

Figure 4-10: RTT variation with increasing Tx power of attack hosts while UAV Tx power is fixed at 50% (or 5W)

of attack hosts and reaches approximately 14 minutes which is obviously, unacceptable. Case I results are as expected because the transmission power of attack hosts was double compared to the regular UAV hosts. Please note that RTT and Tx power denotes the round trip time and transmission power in all graphs of this sub-section.

Figures 4-13 and 4-14 show the results obtained for Case II, where the transmission power of the regular UAV host was double than the transmission power of attack hosts. Once again, the results are very well expected. Since the attack hosts don't have a lot of power to launch the jamming attack with noise signals, they might not be reaching the UAV at all. It is evident from figure 4-13 that the average packet loss stays well below 10% for six or fewer attack hosts and does not reach even 20% even when this number is increased to 14. Similarly, it is clear from figure 4-14 that the RTT suffers and reaches a little over 70 seconds. This value is still under the maximum allowable

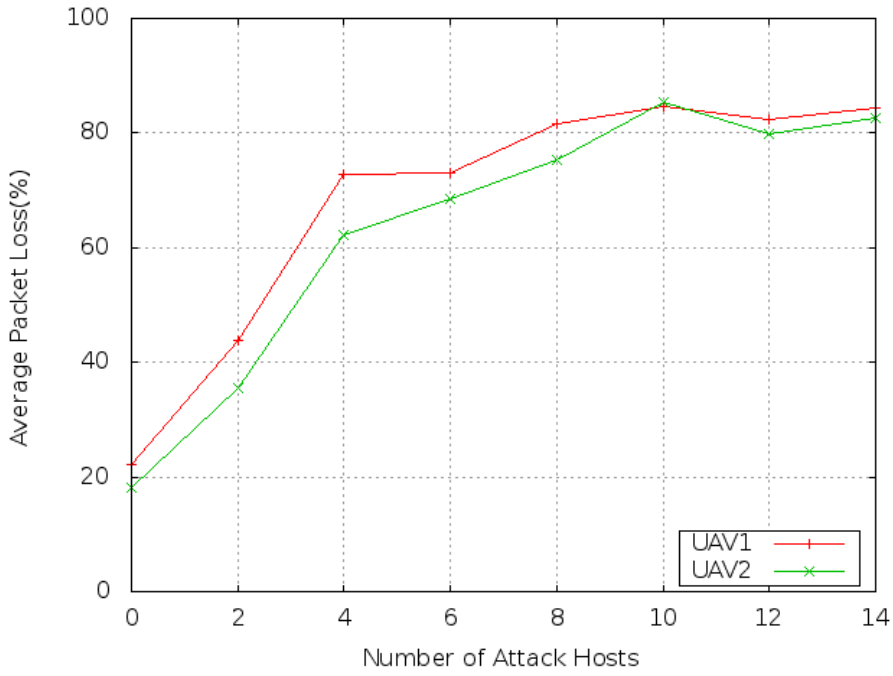

Figure 4-11: Average packet loss for Case I: Host Tx power - 5W and attack host Tx power - 10W

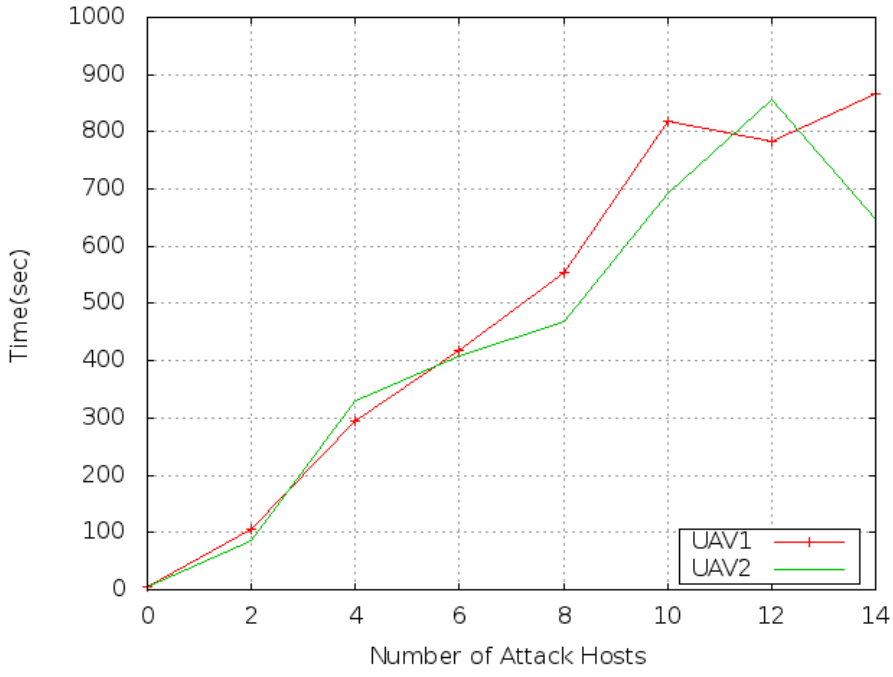

Figure 4-12: RTT for Case I: Host Tx power - 5W and attack host Tx power - 10W

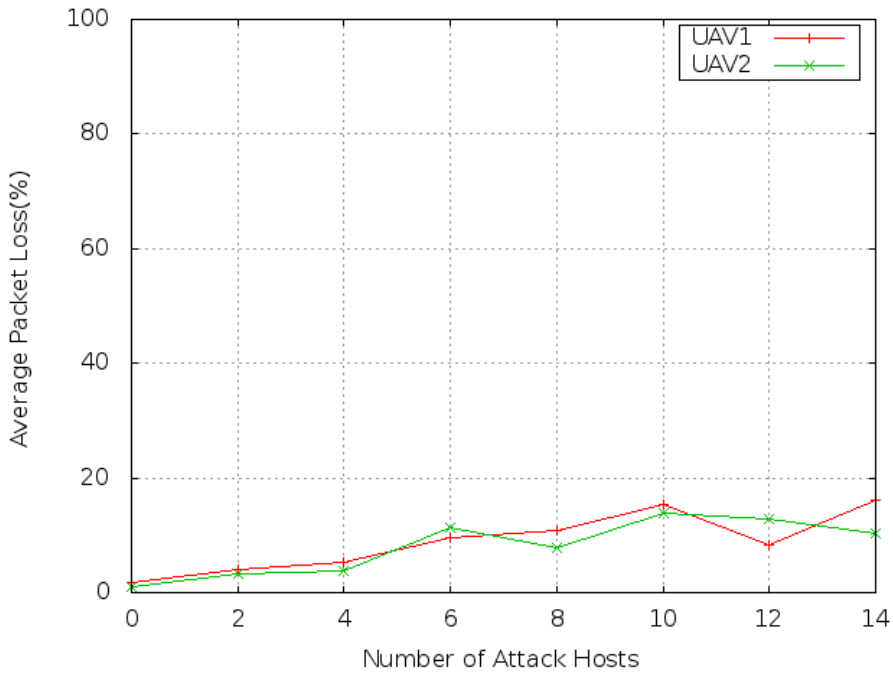

Figure 4-13: Average packet loss for Case II: Host Tx power - 10W and attack host Tx power - 5W

limit of TTL, i.e., 255 seconds. Please note that this value was 800 seconds for worst possible scenario in Case I.

Case III results are shown in figures 4-15 and 4-16. It is quite clear from both the charts that even when no attack hosts were present in the mission area, the packet loss is almost 50%. From Case I and II results, it can be quickly concluded that a transmission power of 2W was not enough for the regular UAVS to communicate. The same loss was 50% in Case I (Host Tx Power: 5W) while it was less than 2% for Case II (Host Tx Power: 10W). A lower transmission power will cause massive losses in any wireless network, and a threshold should be set to establish a working communication link.

Similarly, it is evident from figure 4-16 that even attack hosts don't have much effect on the regular UAVs because of their weak transmission power, except for the case when number of attack hosts was 10. This irregular behavior might have been

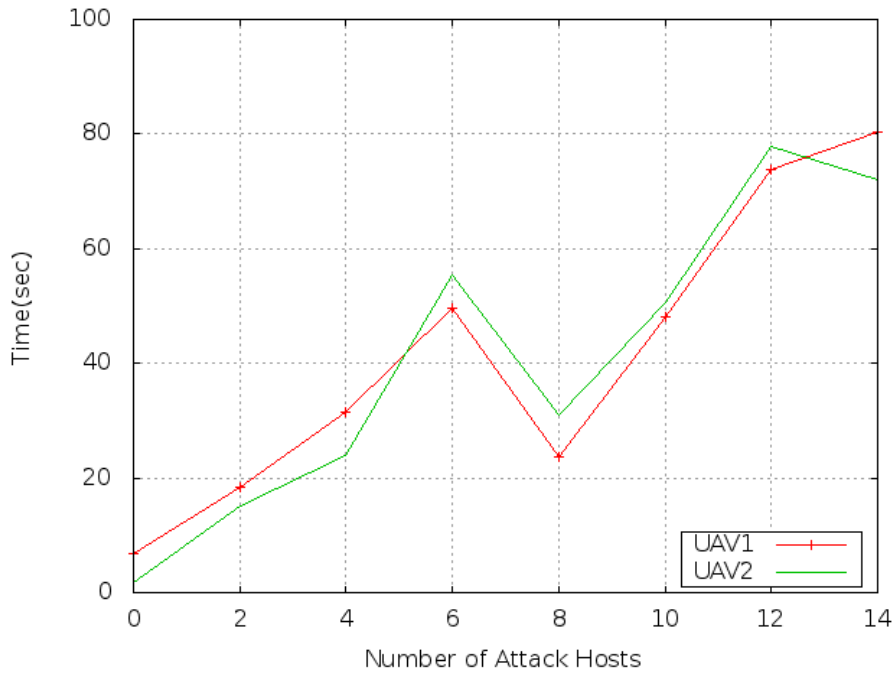

Figure 4-14: RTT for Case II: Host Tx power - 10W and attack host Tx power - 5W

caused because of the random placement of attack hosts in the mission area when the number is changed. Further, this random arrangement may have lead to a closer position to one or more of the regular UAV hosts, which resulted in a successful jamming attack on those particular hosts. Probably because of the same reason, we notice a bump in the figure 4-15 when the number of attack host becomes 10.

4.1.2.2.3 Effect of Simulation Progress

It is beneficial to know the impact of an ongoing attack with time, primarily because such kind of analysis informs the system designers of the degradation in system performance with time. We recorded the average packet loss and RTT for two hosts, host1 (being attacked) and host2 (not being directly attacked) at intervals of 50s to see the trend. Figure 4-17 shows the variation in average packet loss for this evaluation, and it is clear that system performance keeps degrading for both host1

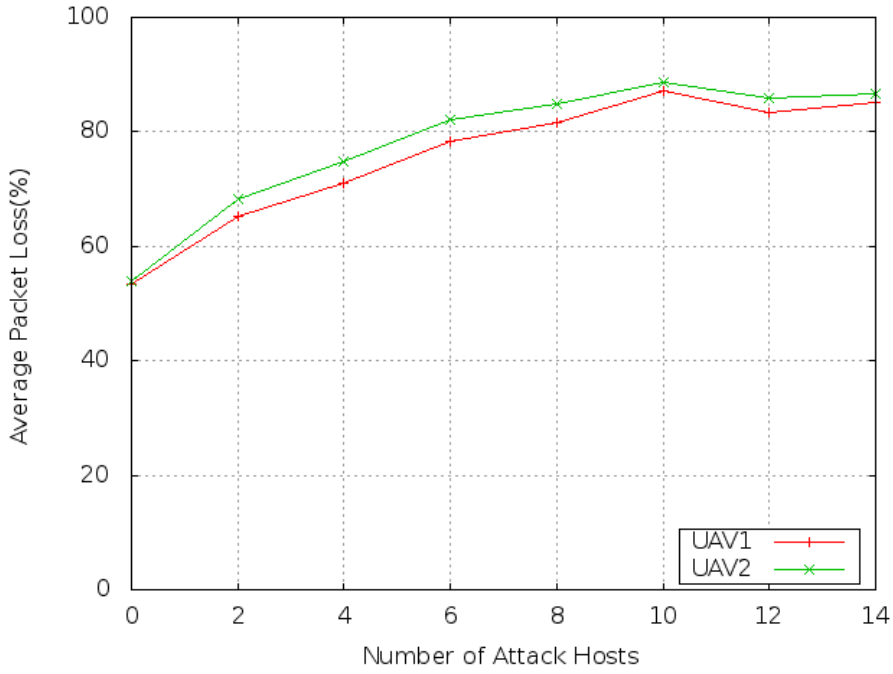

Figure 4-15: Average packet loss for Case III: Host Tx power - 2W and attack host Tx power - 2W

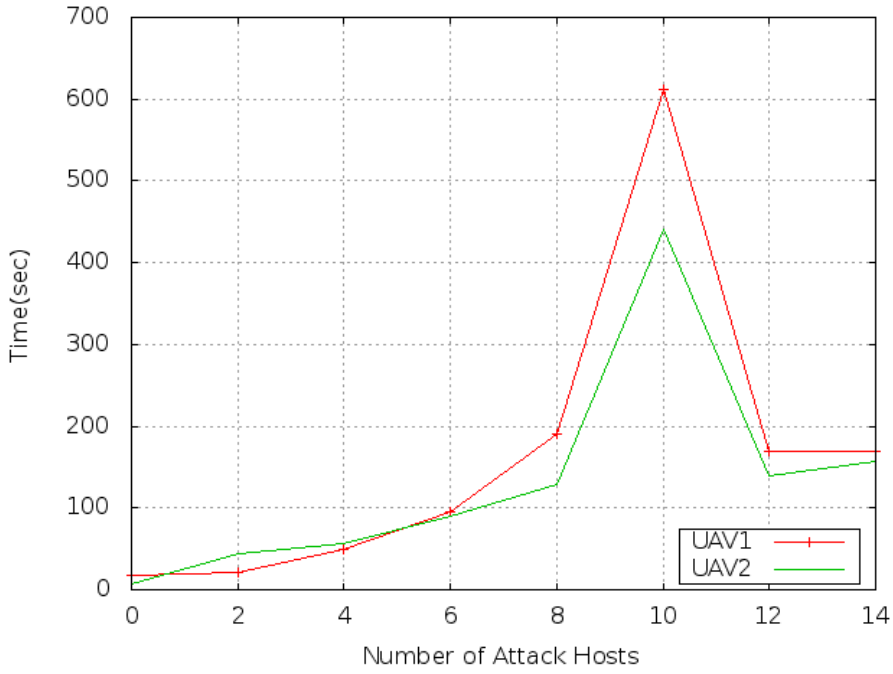

Figure 4-16: RTT for Case III: Host Tx power - 2W and attack host Tx power - 2W

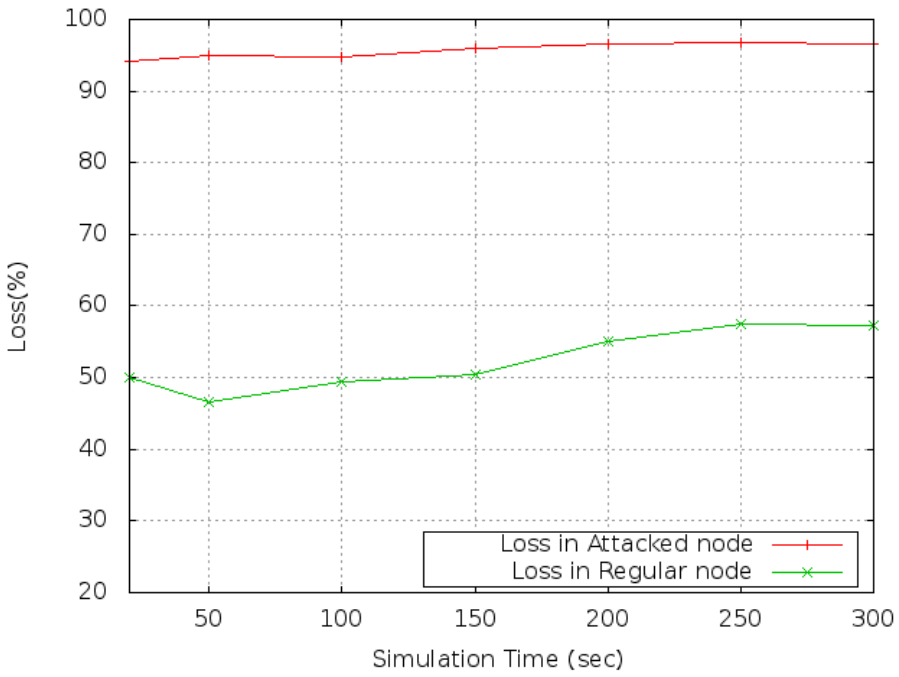

Figure 4-17: Average packet loss variation as simulation progresses to 300s

and host2 as the simulation progresses. It is expected that the behavior would be worse in real world courtesy various other environmental losses. Further, figure 4-18 shows the RTT variation as simulation progress and it is interesting to note that although RTT for attacked host stays above the TTL threshold of 255, it seems to stabilize and remain close to 300. For non-attacked host, the RTT remains close to the median of the $1 - 2$ minute range.

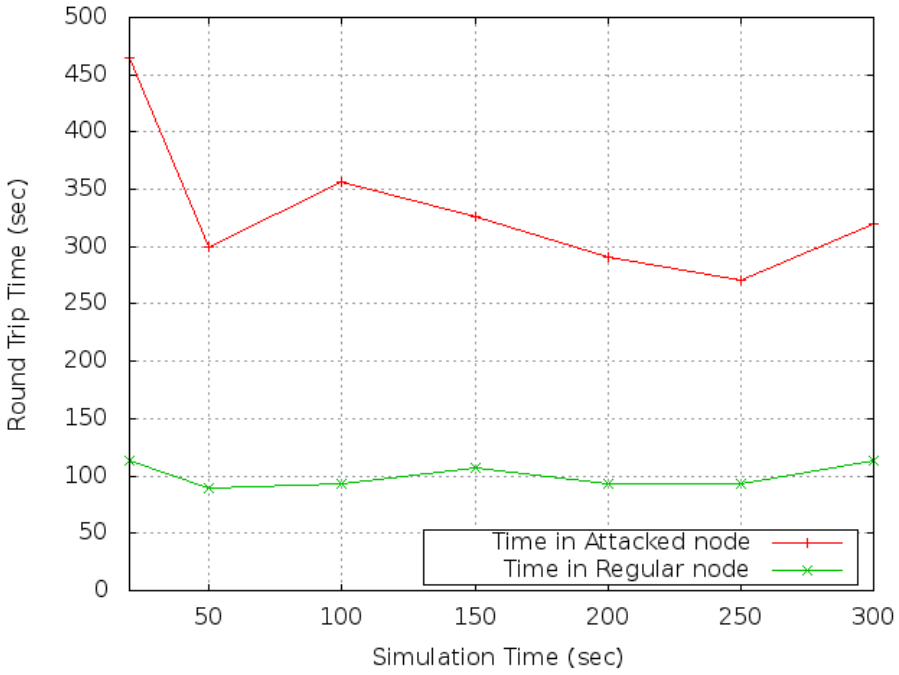

Figure 4-18: RTT variation as simulation progresses to 300s

4.2 Integrity Attacks

As a second class of attacks, we chose attacks that compromised the Integrity of a UAVS or UAVNet. To do so, the attacker needs to have an extensive knowledge about the system they wish to compromise. As a sub-class of these integrity attacks, we decided to implement navigation compromising attacks and chose GPS Spoofing and GPS Jamming attack. This section discusses the design, implementation and related results for these two attacks in our simulation testbed UAVSim.

4.2.1 GPS Spoofing

Studies have established that UAVs are very much vulnerable to GPS Spoofing attack [97–100], and it can cause catastrophic damage to these systems as well as human life. Keeping this in mind, we have implemented this attack and demonstrated

that it can be simulated in UAVSim. One of the most harmful attacks for GPS devices, it aims at spoofing GPS signals to give a false sense of accurate physical location and results in mission path diversion. Nowadays, it is comparatively easier to launch such an attack due to the availability of off-the-shelf GPS signal generators. Satellite constellation preservation and signal transmission precision are of utmost importance in such an attack so that spoofing is not detected. Its implementation has been discussed below in detail.

4.2.1.1 Implementation

We implemented GPS Spoofing attack using a spoofed GPS signal generator, which is, in fact, another vehicle at almost double the altitude of the UAV being attacked. Although civil GPS implementation details are public, building such a generator did pose a few significant challenges due to the complex system design. We assume a higher altitude for attack host to mislead directional antennas installed on a UAV for GPS signal reception. The attack host maintains same angle and distance with the host at all times so that the host does not detect any suspicious activity. The spoofed signal can contain discrepancy in X-coordinates, Y-coordinates or distance as represented in the process flow of the attack implementation shown in figure 4-19.

4.2.1.2 Simulation Results

This section presents various results related to the GPS Spoofing attack. For this attack, the UAV path is important, therefore, two types of mobility, circle and linear, have been used to demonstrate the possible variation through introduction of minor errors in any of position or distance values being sent in a GPS frame.

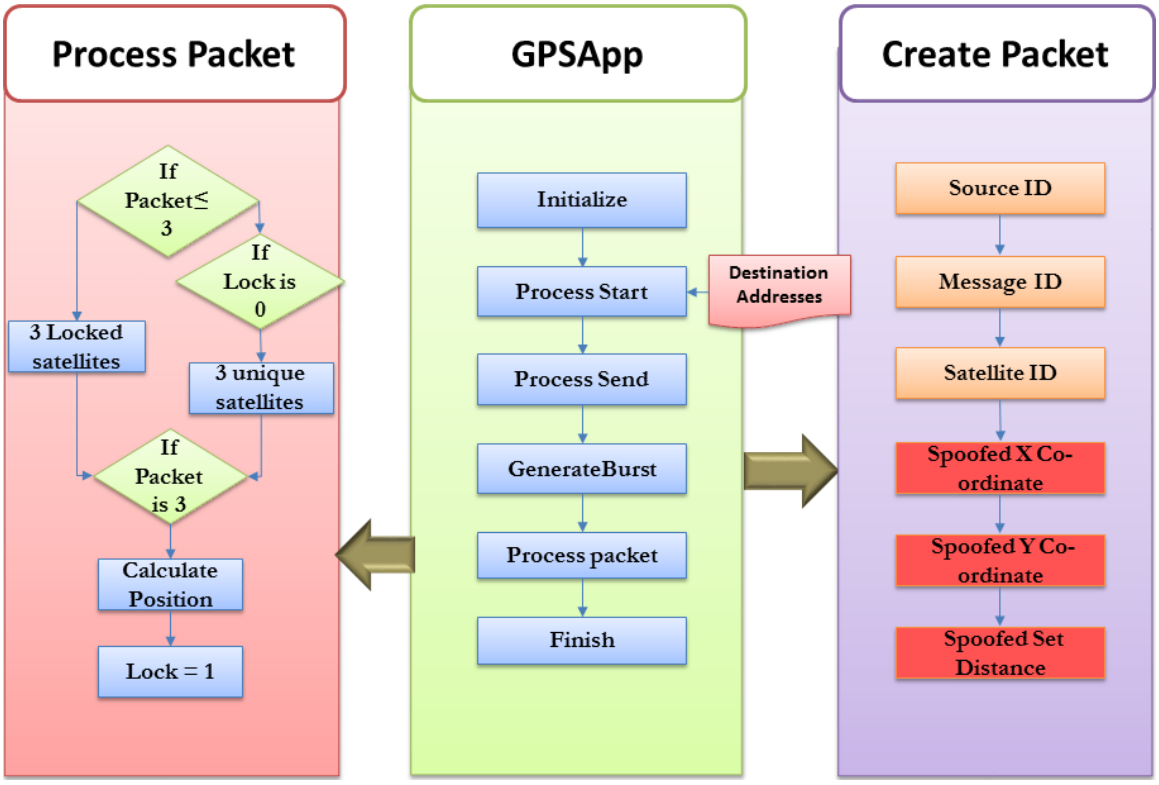

Figure 4-19: Data flow diagram for the GPS spoofing attack implementation

4.2.1.2.1 Effect of GPS Spoofing on Linear path

This sub-section discusses GPS Spoofing attack in UAV Network when the UAV was moving in a linear path. We use a single UAV and a single attack host to launch the attack. Four cases have been evaluated based on the parameter in which the discrepancy was introduced. As mentioned before, we have X, Y and distance values in each GPS packet. Three cases involve discrepancy in each one of them while the fourth case analyzes the effect of discrepancy in all the three parameters.

Case I: Discrepancy in X-direction - In this case, we vary the X-value and keep increasing the discrepancy using the expression

$$x = x + (0.005 * s)$$

Table 4.2: Default satellite parameters

Parameter	Value
Simulation Time Limit	600 seconds
Mobility Type	SatSGP4Mobility
Transmitter Power	500W
Packet Interval	0.5 second
Burst Duration	10 seconds
Sleep Duration	0 second
Position Update Interval	1 second

Table 4.3: Default UAV host parameters

Parameter	Value
Mobility Type	Linear Mobility
Transmitter Power	10W
Speed	250 km/h (0.0037 m/s in simulation)
Burst Duration	10 second
Sleep Duration	0 second
Position Update Interval	1 second

where s is initialized as 0 and incremented by 1 in each new packet generated. Figure 4-20 shows the result of this experiment. As seen in the figure, the original southwest direction of UAV is quite different than the spoofed direction of west. This shows an increase in calculated Y-values while a decrease in calculated X-values.

Case II: Discrepancy in Y-direction - In this case, we vary the Y-value and keep increasing the discrepancy using the expression

$$y = y + (0.005 * s)$$

where s is initialized as 0 and incremented by 1 in each new packet generated. Figure 4-21 shows the result of this experiment. As seen in the figure, the (almost) west direction of UAV is the actual path while spoofed GPS makes the UAV think that it is going in the northwest direction. This shows a huge decrease in Y-values while a very minimal impact on X-values comparatively. Clearly, the angle of variation will

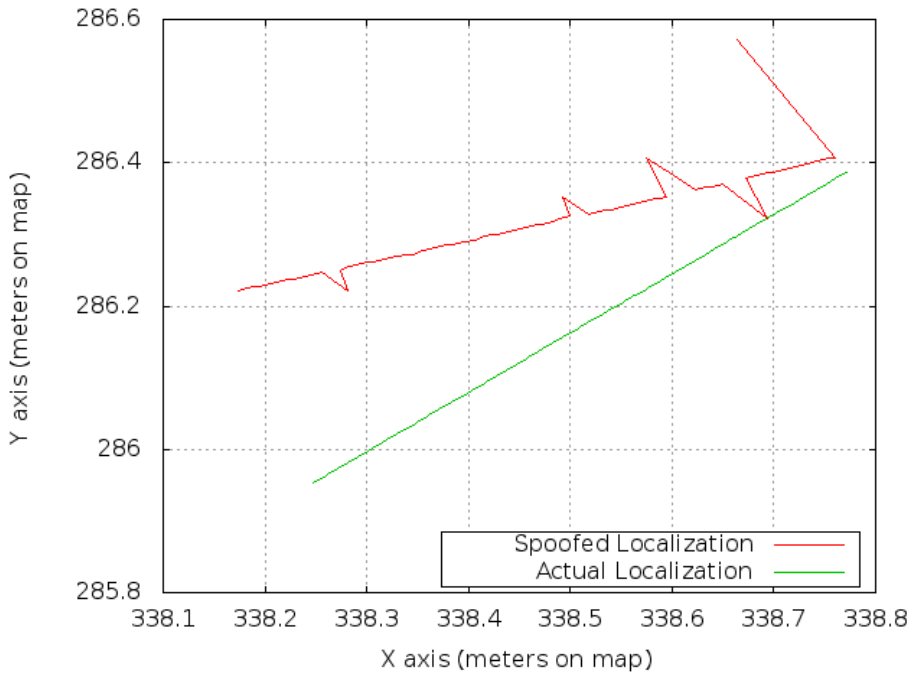

Figure 4-20: Effect of X-value discrepancy on linear path of the UAV

increase if we increase the discrepancy factor of 0.005.

Case III: Discrepancy in X and Y-direction - In this case, we vary the Y-value and keep increasing the discrepancy using the similar expressions of Case I and II. Figure 4-22 shows the result of this experiment. As seen in the figure, the UAV thinks that it is moving in almost reverse of its actual direction. This shows that both X and Y values are now increasing very rapidly.

Case IV: Discrepancy in X, Y and Distance - In this case, a similar expression is used to introduce a discrepancy in all the three variables of X, Y and the distance. Figure 4-23 shows the result of this experiment. Such discrepancy introduction shows that the spoofed path is similar to the one obtained when discrepancy was introduced only in Y-values. This indicates that discrepancy in distance values somewhat negates the effect of discrepancy in X-values.

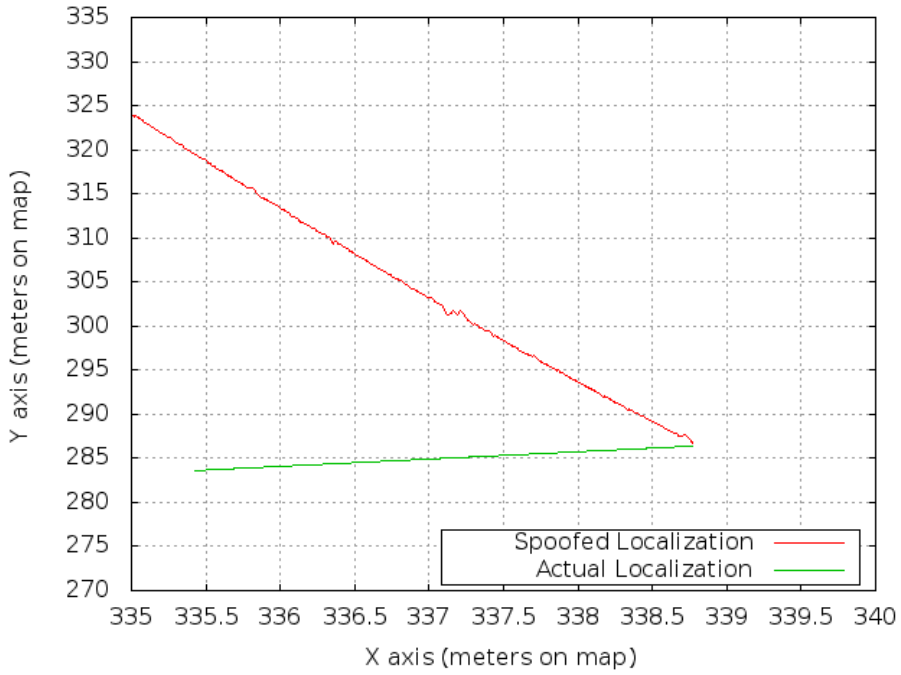

Figure 4-21: Effect of Y-value discrepancy on linear path of the UAV

4.2.1.2.2 Effect of GPS Spoofing on Circular path

In a second set of experiments, the GPS spoofing attack was carried out on a host moving in a circular path. Its initial position can be anywhere on a circular path with a radius of 1m and center (561m, 432m) on the map. The starting position was selected randomly to introduce randomness of UAV position and see if results were location independent. The attack host also moves in a circular path with its starting position on a circular path of radius 2m and center (565m, 435m). The attacks were designed considering different data broadcast from the attack host. Five cases were analyzed for this particular scenario that are different from linear path scenarios.

Case I: Discrepancy in X-direction - In this case, a discrepancy s is added to X-values with a factor of 0.005 using the expression

$$x = x + (0.005 * s)$$

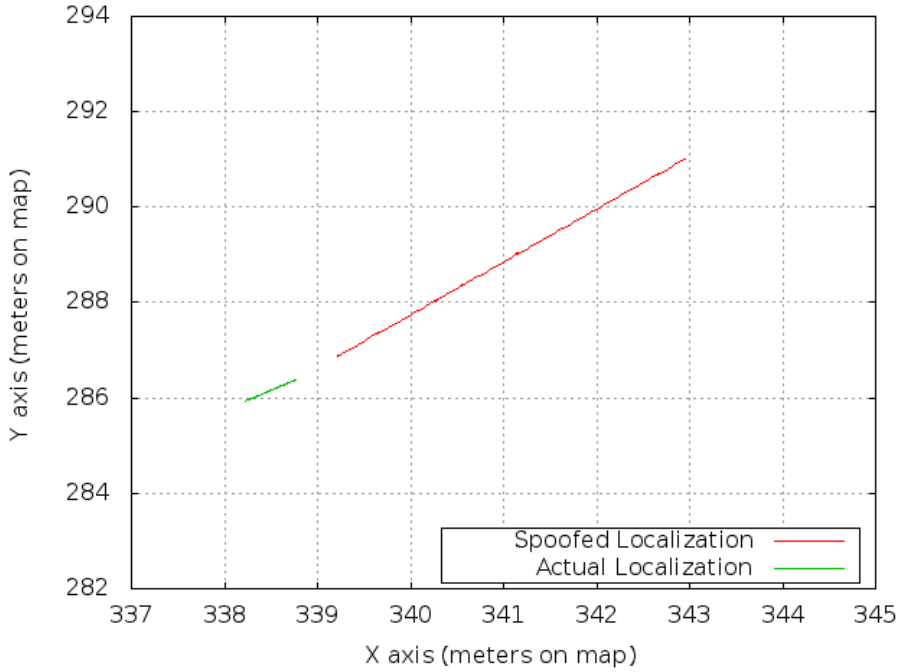

Figure 4-22: Effect of both X and Y-value discrepancy on linear path of the UAV

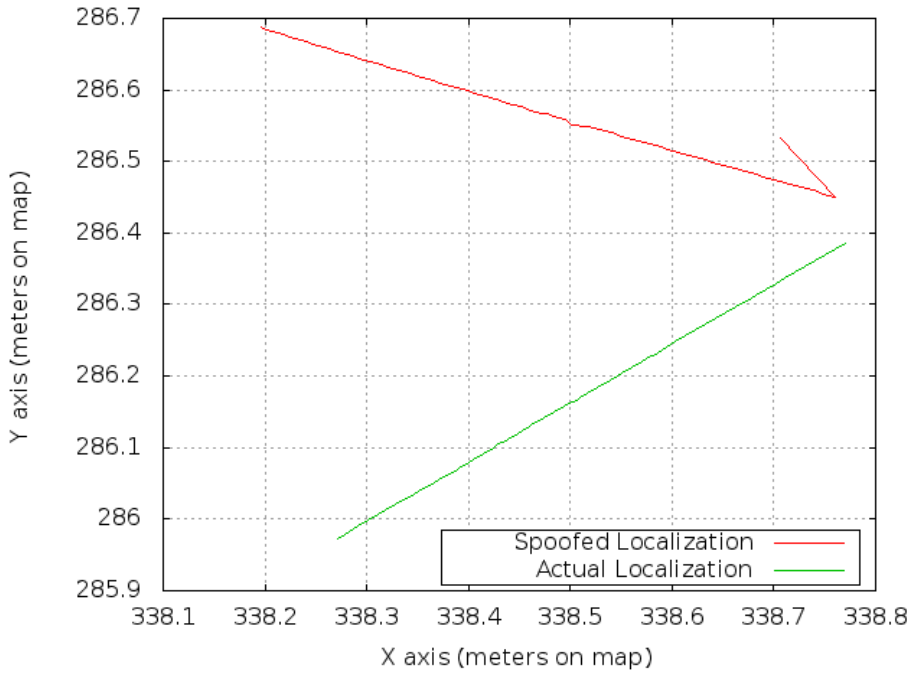

Figure 4-23: Effect of distance, X and Y-value discrepancy on linear path of the UAV

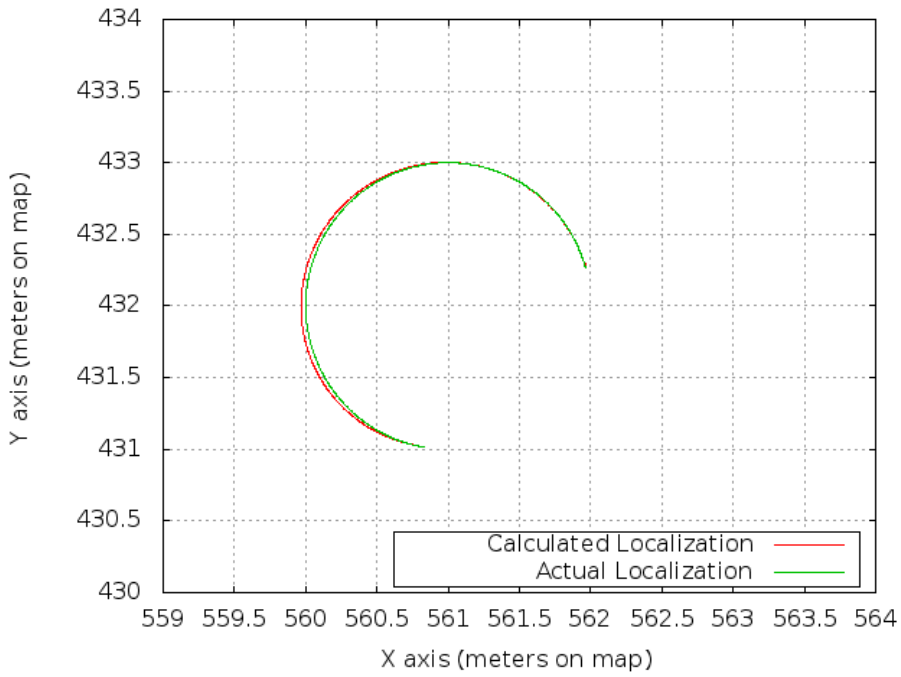

Figure 4-24: Effect of low X-value discrepancy on circular path

where, s is initialized as 10 and incremented by one as each new packet is generated. Figure 4-24 shows the obtained result for this experiment. It is clear that there was a minor deviation of the host from its original circular path and it almost traverses the same original path.

Case II: Higher discrepancy in X-direction - Since increasing the discrepancy factor little by little was not resulting in tangible changes, we increased the discrepancy factor in X-values by three times to 0.015 while keeping Y and distance values same for this case. Similar to the Case I, s was initialized as 10 and incremented by one as each new packet is generated. Figure 4-25 shows the result obtained for this case. As shown, the spoofed path is quite different from the original path and becomes linear from the original starting point in the opposite direction.

Case III: Positive discrepancy in X and Y-directions - In this case, we introduce positive discrepancy in both X and Y-values using similar expression as Case I. Result for this case is shown in figure 4-26 and it shows that the host is actually moving

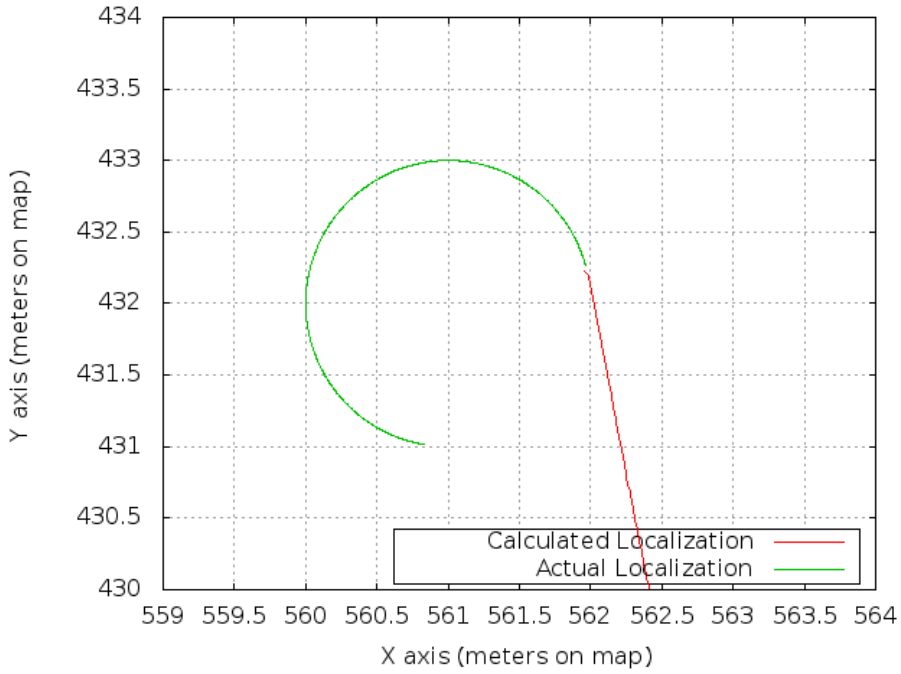

Figure 4-25: Effect of high X-value discrepancy on circular path

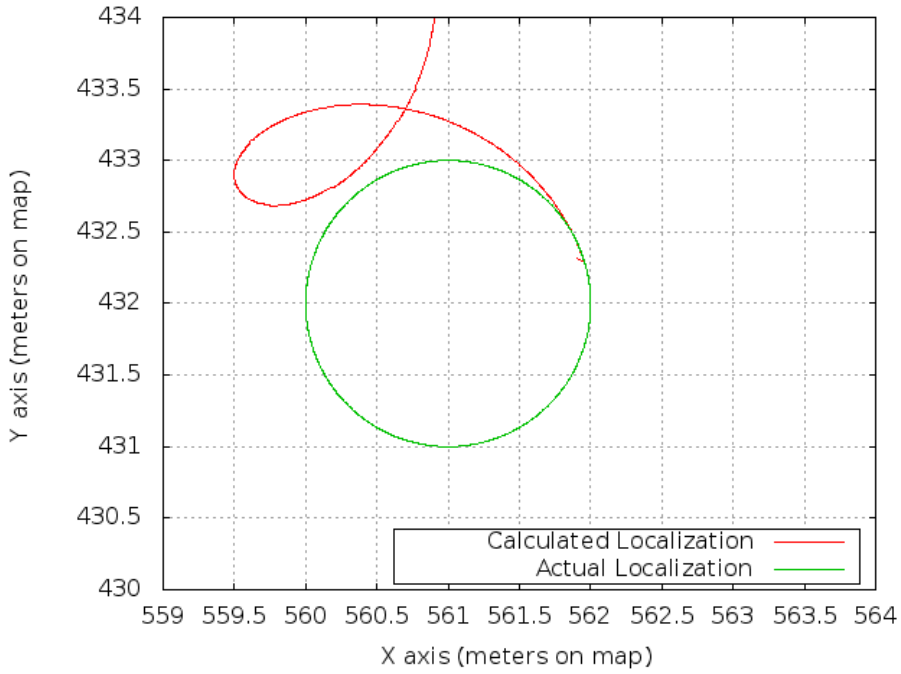

Figure 4-26: Effect of positive X and Y-value discrepancy on circular path

outward in a helical path with varying pitch, while believing that it is moving in a circular path. It should be noted that the variation is mostly increasing Y-values and thus, results in a helical path.

Case IV: Negative discrepancy in X and Y-directions - This case involves negative discrepancy introduction in both X and Y-values using similar expression as Case I. Related result are shown in figure 4-27 which clearly shows that initially, the host followed a path close to the original path and then moved outward following a modified helical paths. It should be noted that such a discrepancy is resulting in large negative variations in both X and Y-values.

Case V: Positive discrepancy in X-direction and negative in Y-direction - In this case, discrepancy was added to X-values while subtracted from Y-values. Figure 4-28 shows the results obtained for this case. It can be seen that the host follows inward helical path moving away from its original position. This kind of discrepancy is resulting in lower positive variation in Y-values while higher, or almost double positive variation in X-values, and this results in such a helical path.

4.2.2 GPS Jamming

A famous and well known attack which affects the availability of GPS Navigation and causes total communication failure. As mentioned earlier, Jamming involves transmission of high and/or low power noise signals to render all communication receivers in the area non-functional. Several anti-jamming techniques have been proposed to prevent narrow band interference as well as wide band interference, but there is still no absolute solution to this problem. In this subsection, we discuss the implementation of GPS Jamming.

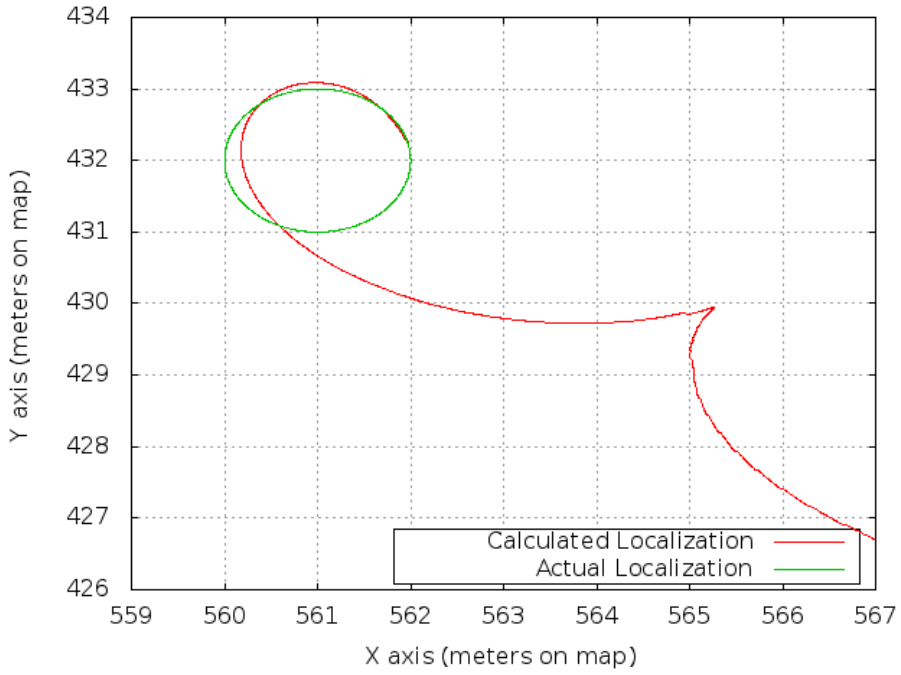

Figure 4-27: Effect of negative X and Y-value discrepancy on circular path

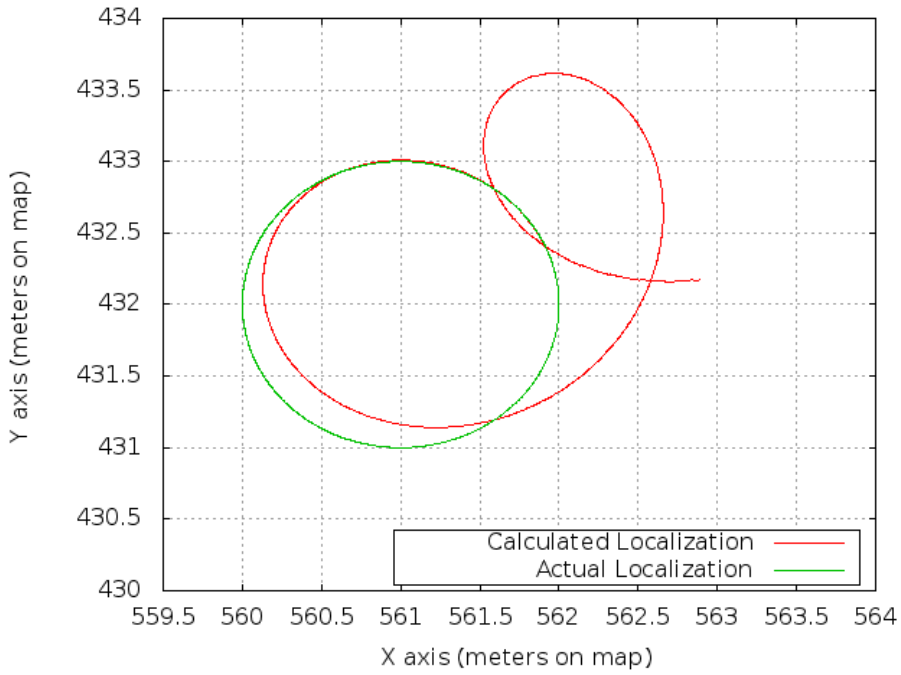

Figure 4-28: Effect of positive X-value discrepancy and negative Y-value discrepancy on circular path

4.2.2.1 Implementation

We implemented this attack using a large number of attack hosts that roam around in the mission area and keep transmitting noise signal with high data rate and power to jam all frequencies. Jammer implementation in our testbed was abstracted through the use of several hosts to block each frequency intended to be jammed. This abstraction is being used due to the limitation of the underlying simulation engine. Clearly, to jam all communication frequencies, the number of attack hosts required will be quite large. Therefore, we jammed only a few GHz of bandwidth based on the frequencies being used in the target area.

For GPS Jamming attack, we have only used linear mobility. In all cases, the speed on the map was 0.0037 meters per second. As mentioned earlier, this corresponds to a speed of 250 km/h in the real world. Other default parameters for satellite are shown in table 4.2 and for hosts are shown in table 4.3. As mentioned earlier, GPS Jamming attack was implemented in our testbed by using different attack host to jam different frequencies. Therefore, it is understood that a certain number of attacks hosts would be required to jam a particular frequency range. This results in partial blockage of the frequency range and thus, a lesser packet loss of GPS data. Nonetheless, as discussed later, the attacks were successful.

4.2.2.2 Simulation Results

For the GPS jamming attack, we used 10 UAV hosts roaming in the map in an area of $1000 * 1000$ km. Attack hosts are roaming in the area as well, transmitting noise signals without any knowledge of UAVs in that area. This specific scenario makes it more real. We varied the number of attack hosts from 0 to 20 to check the behavior of the network. The plotted results are shown in figure 4-29 where we show the average packet loss for these 10 hosts for each case. As expected, the loss increases

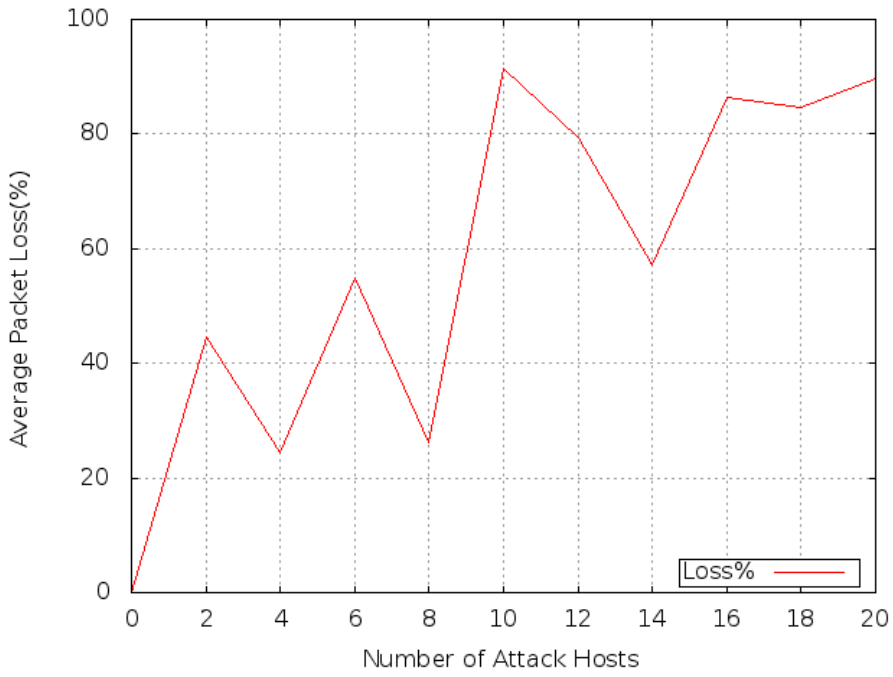

Figure 4-29: Average GPS packet loss with increasing number of attack hosts for 10 UAV hosts

with increasing number of attack hosts. Some of the lower losses may represent the unknown movement of the UAVs as well as attack hosts in the target area.

4.3 Attack Defense

In this section, we discuss an attack mitigation and recovery measure developed in UAVSim to prove its capability to test various defensive measures and evaluate their performance. To this end, we have implemented a well-known GPS Spoofing detection and mitigation technique called RAIM (Receiver Autonomous Integrity Monitoring). Although various RAIM implementations are available in the literature [101, 102], we attempted to design our independent implementation and test its performance. A brief introduction about RAIM, its implementation and related results are presented in this section.

4.3.1 RAIM

Receiver autonomous integrity monitoring (RAIM) was developed as a technique to improve the accuracy and reliability of GPS through the use of algorithms adapted to GPS signal architecture. Assuming that there is at most one type of error present at a time, there are two aspects of testing possible, namely, the detection test and the correction test [101]. A detection test is used to identify the exact error while the correction test corrects it and tests if the correction is working. In the previous section, it was observed that the Y-value discrepancy often produced large position errors and may result in catastrophic damages. Therefore, we implement these two phases in our algorithm.

4.3.1.1 Design and Implementation

The figure 4-30 shows the process flow of the algorithm. Our RAIM algorithm compares the pseudo-range measurements among themselves to ensure that they are all consistent. The primary flow of any RAIM algorithm would be to apply fault detection mechanism on the computed set of the navigation solution, isolate faulty satellites and provide mitigation level computation so that its availability is maintained [103]. Since 3D position calculation requires data from four satellites, four visible satellites are not enough to provide integrity. If five visible satellites are chosen and an anomaly is detected, then position calculation from that satellite is discarded. The rest of the four satellites, again, are not enough to compute the location using different measurements and verify that the solution is indeed correct. Therefore, the receiver can issue a warning but not provide integrity. With 6six or more satellites, the receiver can detect and mitigate the impact of faulty satellite [103].

A similar method has been used in our GPS anti-spoofing (RAIM) algorithm design where five satellites are used for detection and mitigation based on our 2D

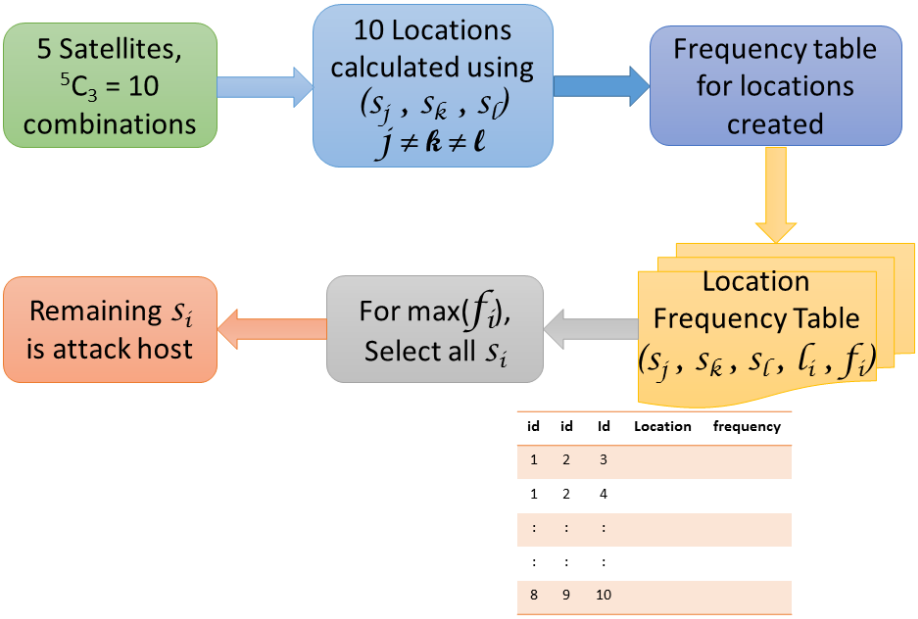

Figure 4-30: RAIM implementation algorithm flow

implementation constraint. When the packets are received for the first time, there is no lock on the satellites. To implement RAIM, the original GPS implementation algorithm was modified to accept packets from five unique satellites instead of three. Once the packets are received, RAIM algorithm is called to detect any attack host. Various combinations of these satellites, which will be 10, and their corresponding packet data are created, and position of the host for each combination is calculated through the trilateration equations. Clearly, if the data in all packets are correct, the position of the host calculated by each combination of satellites should be approximately equal. If there is the slightest error in the data sent by the satellites, it can be identified up to 2 digits after the decimal. After the threshold of the computational differences, the discrepancy becomes apparent and indicates that one of the satellites in that combination must be an attack host. Other combinations are checked for the discrepancy and the satellite id common in all those combinations is labeled as the attack host.

Once the attack host is identified, three out of the remaining four satellites are

locked, and the host starts accepting packets from them. It is possible that the discrepancy in packets sent by the attack host is not detected in the first attempt, and the host acquires a lock on it. The idea of the spoofing attack is to deviate the path of the host. The RAIM function is called repeatedly at fixed intervals to check if the calculation was indeed correct. The next time RAIM is called, the discrepancy in the position calculated would naturally increase, and it will be easily identifiable, and thus, the mitigation technique could be successful. From the results, if three or more satellites appear to be faulty, the lock is completely released and a new set of satellites are selected which go through the same detection and mitigation steps before the lock is acquired again.

4.3.1.2 Simulation Results

To test accurate working of our implementation of RAIM and check various parameters affecting its affect, we ran various simulations while introducing discrepancy in X as well as Y-values of the coordinates. Following are various results obtained through these experiments.

4.3.1.2.1 X-value Discrepancy Detection and Correction

First, a discrepancy was introduced in the X-values of the transmitted coordinates. RAIM duration is a parameter that sets the time interval after which RAIM will run again to check the consistency of the position calculated. We varied the RAIM duration from 30 seconds to 120 seconds, and it was observed that RAIM works best when this interval of the consistency check is set at 90 seconds. Please note that one unit on these graphs is about 9.25 km, and the speed of the UAV was 250 km/h.

Figures 4-31 to 4-35 show the obtained results for X-value discrepancy introduction detection and correction for the durations as mentioned above. Please note that the green line indicates the actual path, the red indicates the spoofed path while pink

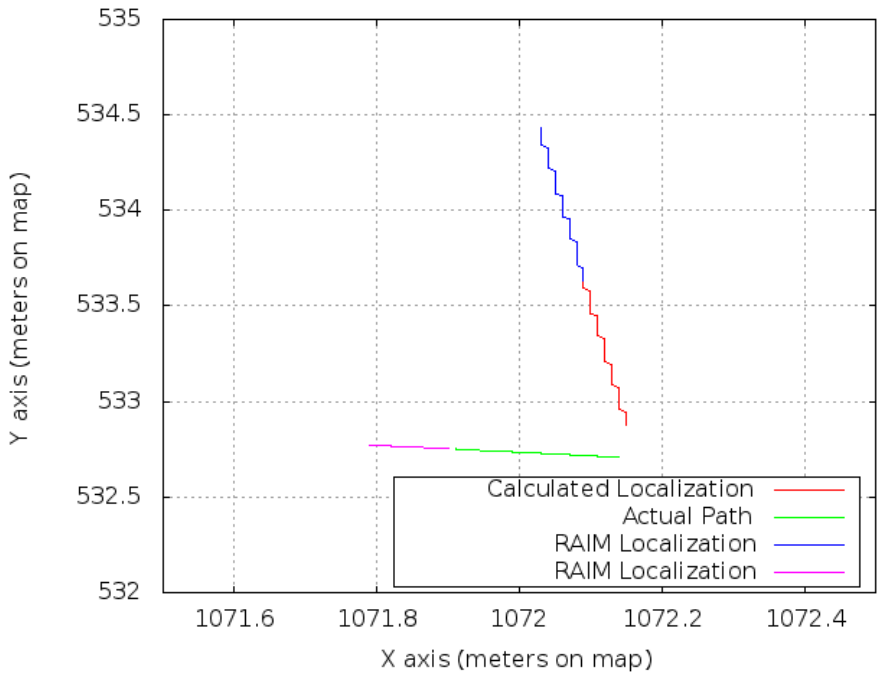

Figure 4-31: X-value discrepancy detection and correction for RAIM duration=30s

and blue indicates the RAIM corrected path. It is evident from these figures that the distance (or time) taken by the RAIM algorithm to detect the GPS-spoofing attack is quite high for shorter RAIM durations. Up to a certain RAIM duration (90 seconds), the detection distance (or time) decreases but we see that when the RAIM duration is increased to 120 seconds, its performance reduces drastically. Figures 4-34 and 4-35 clearly indicate this reduction in RAIM performance and show the correction distance for 90 and 120 seconds, respectively. Readers should also note that the direction of the spoofed path is variable in each simulation. This variation is due to the random initial location of the attack host in each simulation that results in path change in a different direction each time.

Interestingly, one of the shortcomings of our RAIM implementation was noted to be its inability to correct the path when RAIM duration was too short, i.e., 30 seconds. As evident from figure 4-32, although RAIM was able to detect a discrepancy, initial

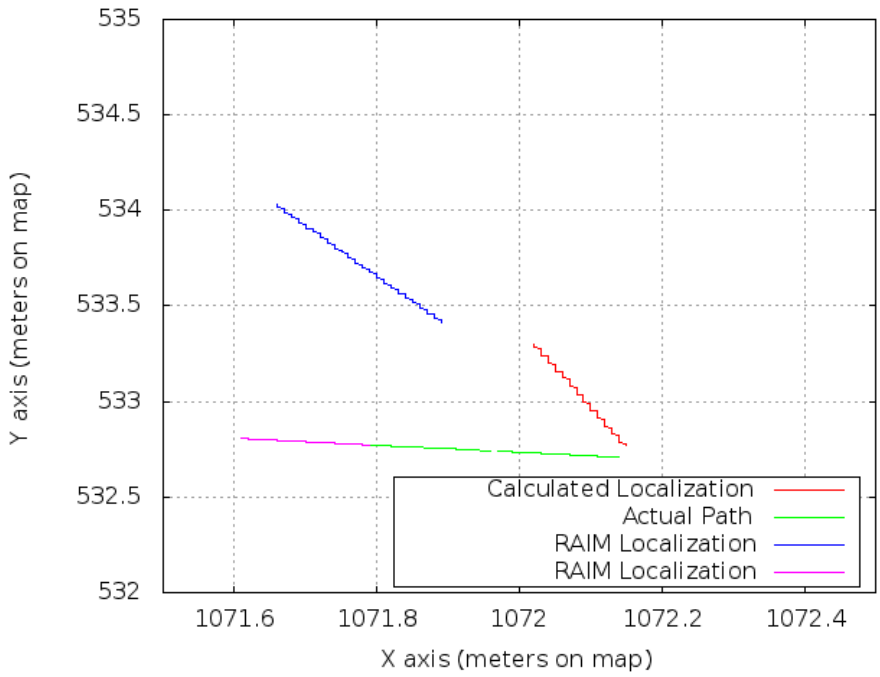

Figure 4-32: X-value discrepancy detection and correction for RAIM duration=45s

RAIM correction (blue) was still unable to correct the path, and it was continuing calculating the incorrect path. This means that the correction algorithm was not able to differentiate the attack host from genuine navigation satellites. In the second iteration after the RAIM duration gap, RAIM was able to correct the path once the attack host introducing discrepancy was easily identifiable due to significant errors in localization. Further, the correction was improved once this duration was increased to 45 seconds as shown in figure 4-33. In this case, the distance (or time) taken by RAIM algorithm to autocorrect the position information was larger than the 30s case. When the RAIM duration is further increased to 60 seconds, the correction was accurate but the distance taken was too large.

When this duration is increased to 90 seconds, the detection and correction was very accurate as well as quick. Finally, when the RAIM duration is further increased to 120 seconds, we see the adverse effects on the RAIM operation. Although the

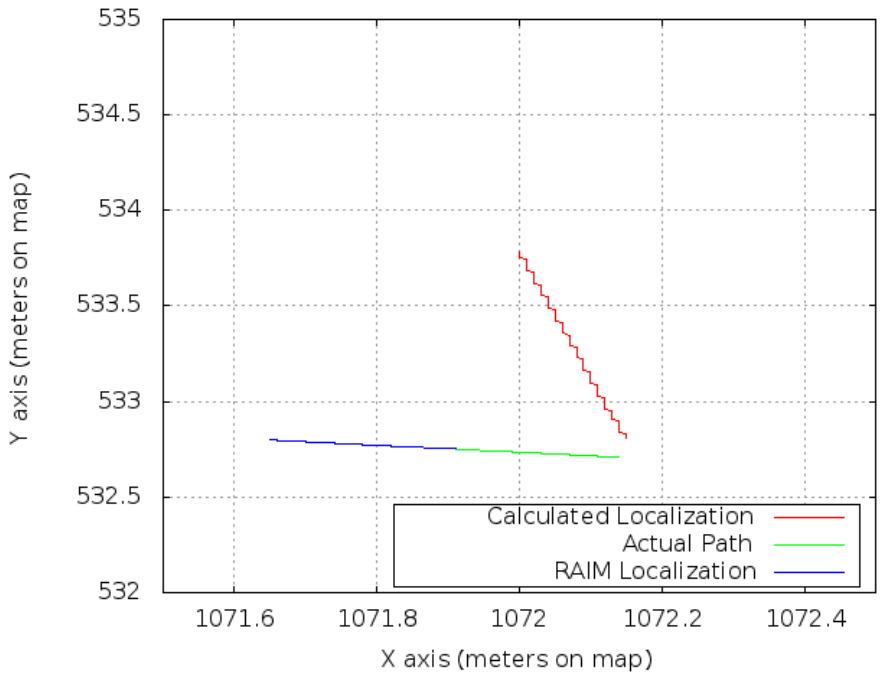

Figure 4-33: X-value discrepancy detection and correction for RAIM duration=60s

correction is accurate in this case, it takes almost 3 times the distance than the 90 seconds case to correct the path.

4.3.1.2.2 Y-value Discrepancy Detection and Correction

Next, a similar discrepancy was introduced in the Y-values of the transmitted coordinates. Similar to the previous experiment, we varied the RAIM duration from 30 seconds to 120 seconds, and it was once again observed that RAIM works best when this interval of the consistency check is set at 90 seconds. Thus, the RAIM duration was used as 90 seconds for all other RAIM related simulations. We will now discuss these results in detail.

Figures 4-36 to 4-40 show the obtained results for Y-value discrepancy introduction detection and correction for the durations mentioned above. Once a comparison is made between the results for X-value discrepancy and the Y-value discrepancy, it is

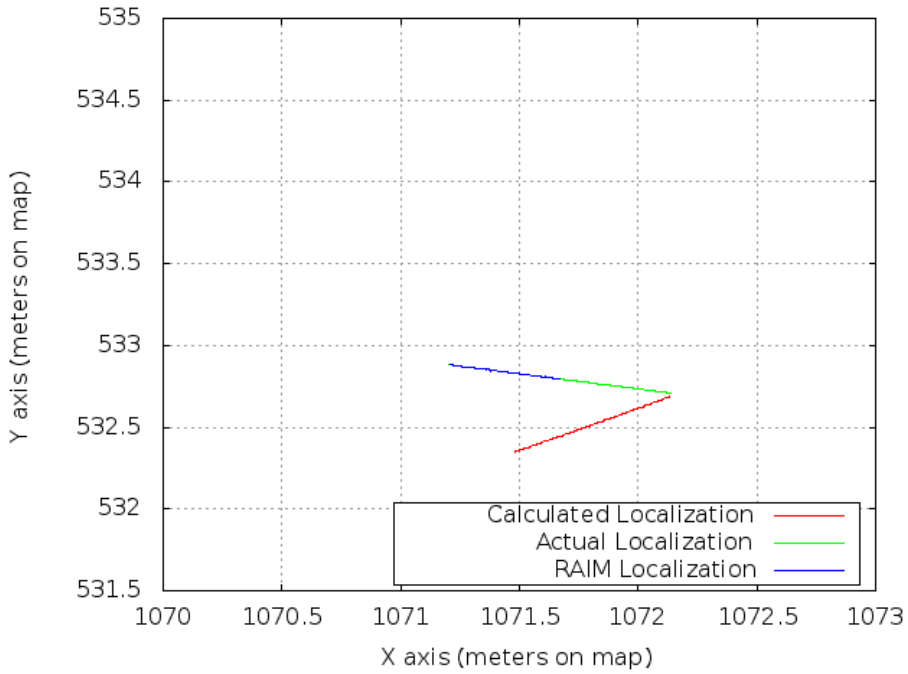

Figure 4-34: X-value discrepancy detection and correction for RAIM duration=90s

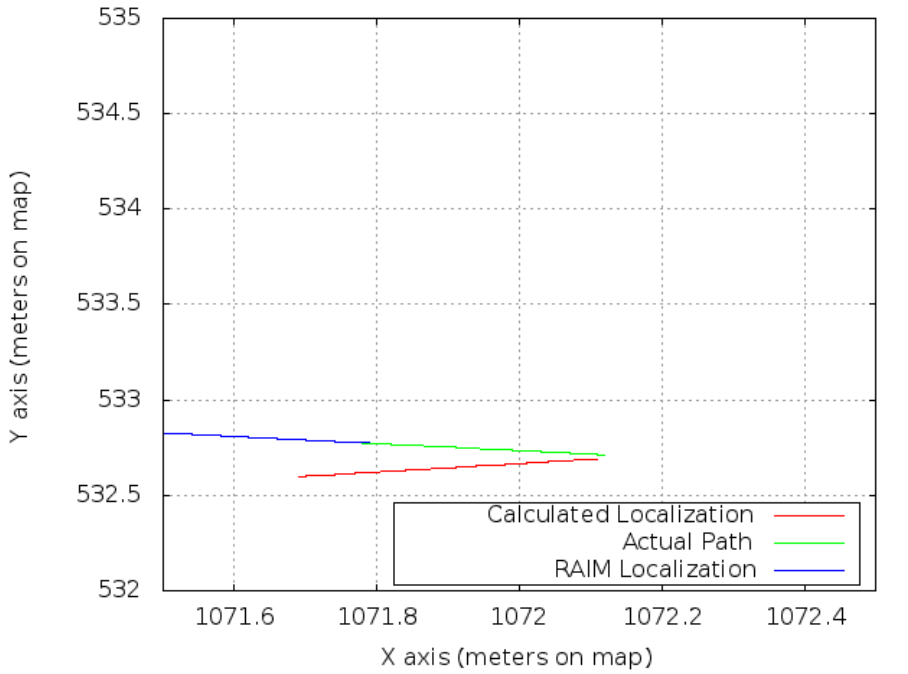

Figure 4-35: X-value discrepancy detection and correction for RAIM duration=120s

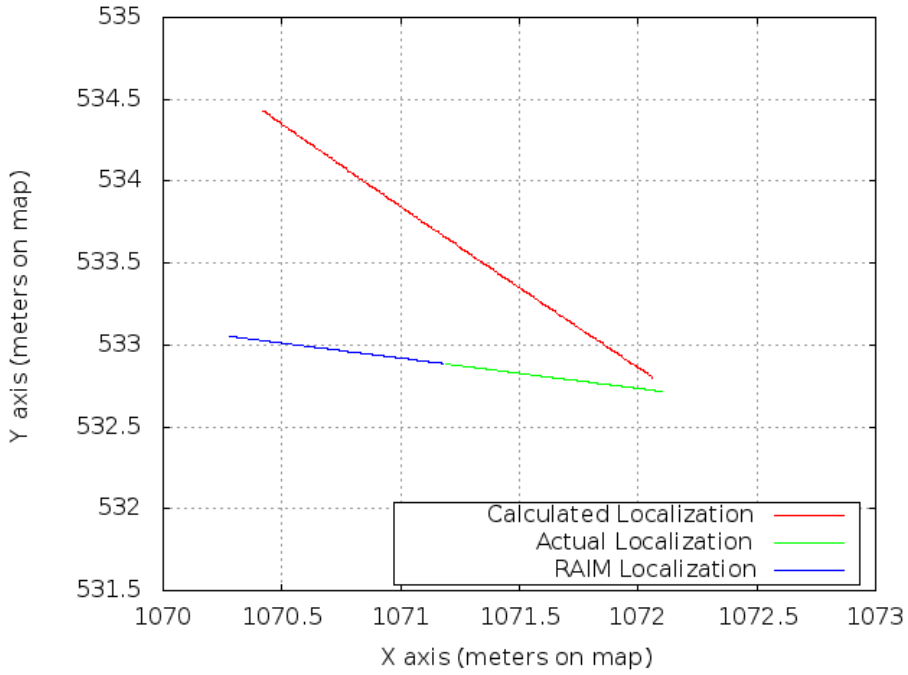

Figure 4-36: Y-value discrepancy detection and correction for RAIM duration=30s

notable that the correction works better for the Y-value discrepancies. At this point of time, our earlier analysis of GPS spoofing attack should be revisited. According to those results, Y-value discrepancies were much worse compared to X-values and thus, should be easily detectable and explains why the performance for RAIM is better for Y-value discrepancies.

As indicated in Figures 4-36 and 4-37, for RAIM durations of 30 and 40 seconds, the results are quite different for this evaluation. RAIM is not only able to detect but accurately correct the path of the attacked host. The only issue here is the longer distance (or time) it takes to make that correction. Similar to previous X-value discrepancy results, the detection was improved for other RAIM durations until a duration of 90 seconds. Finally, as evident from figure 4-40, the performance at a duration of 120 seconds was almost three times worse than the best case of 90 seconds.

As a conclusion, it can be noted that the overall performance of our RAIM imple-

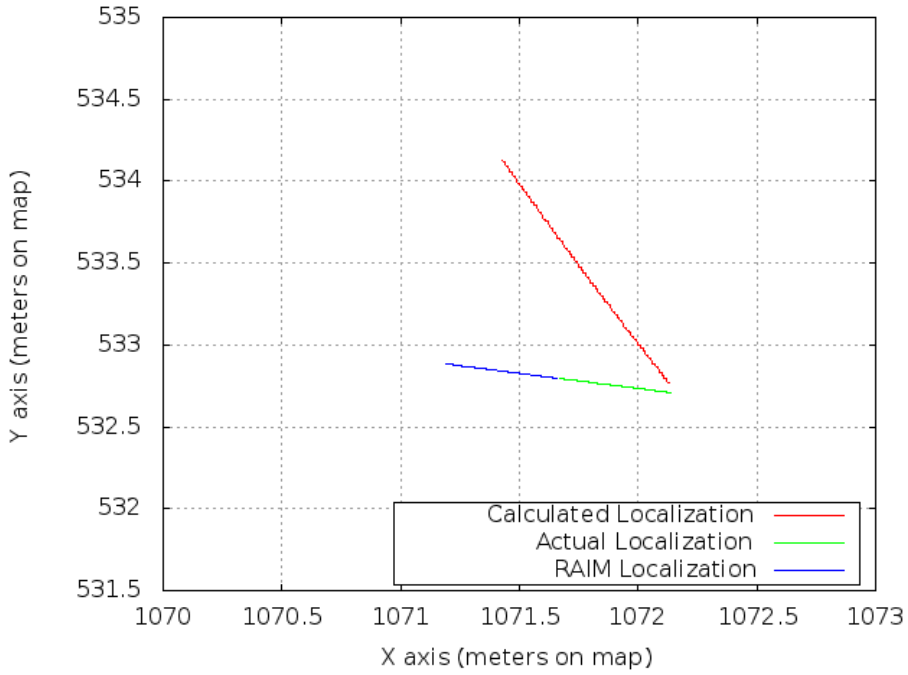

Figure 4-37: Y-value discrepancy detection and correction for RAIM duration=45s

mentation is acceptable and it can correct the path of the UAV within few kilometers of the launch of attack when the RAIM consistency check is set as 90 seconds. Further improvement of this algorithm for better and quicker detection as well as correction can be undertaken as future work.

4.4 Chapter Summary

Through all these attack implementation, various results were obtained and valuable insights were gained. These insights further prove the usability and close-to-real simulation capability of the testbed. Some of these insights are listed below:

- Most cyber-attacks, which aim to take control over the subject, do not involve a large amount of data transmission. Instead, these attacks require minimal data transfer of some unique command and control messages. Therefore, sim-

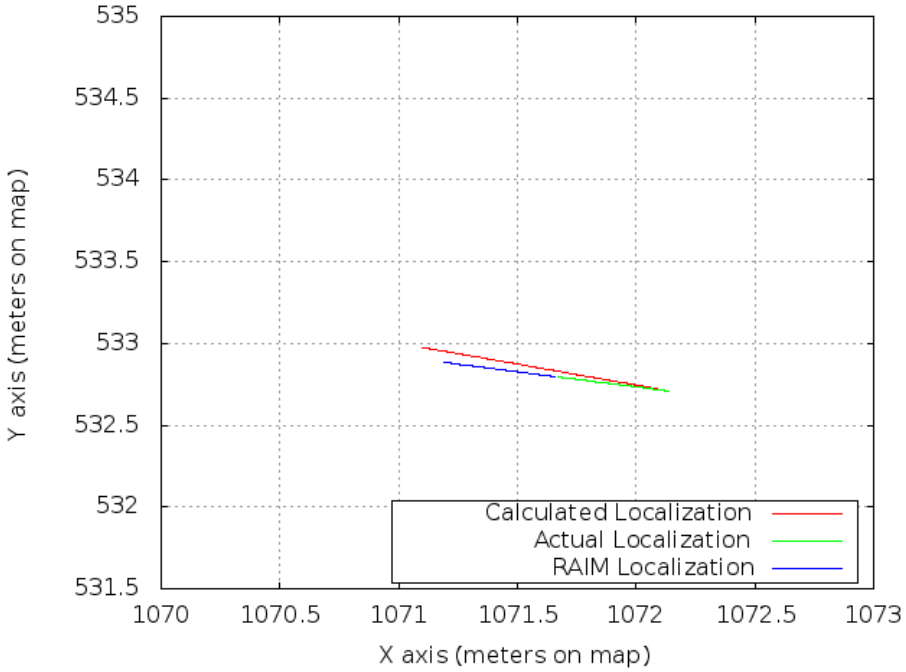

Figure 4-38: Y-value discrepancy detection and correction for RAIM duration=60s

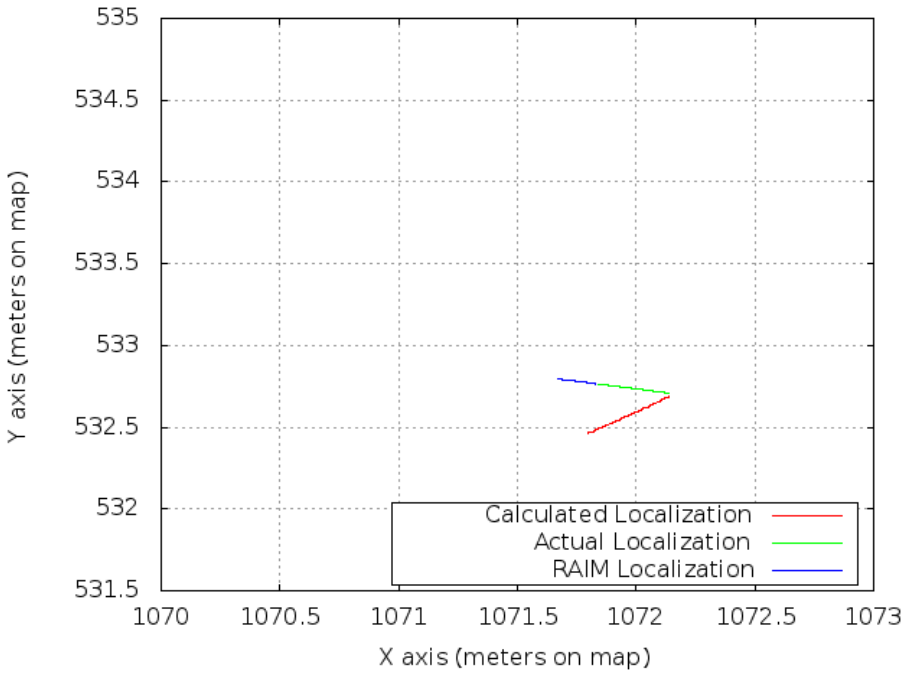

Figure 4-39: Y-value discrepancy detection and correction for RAIM duration=90s

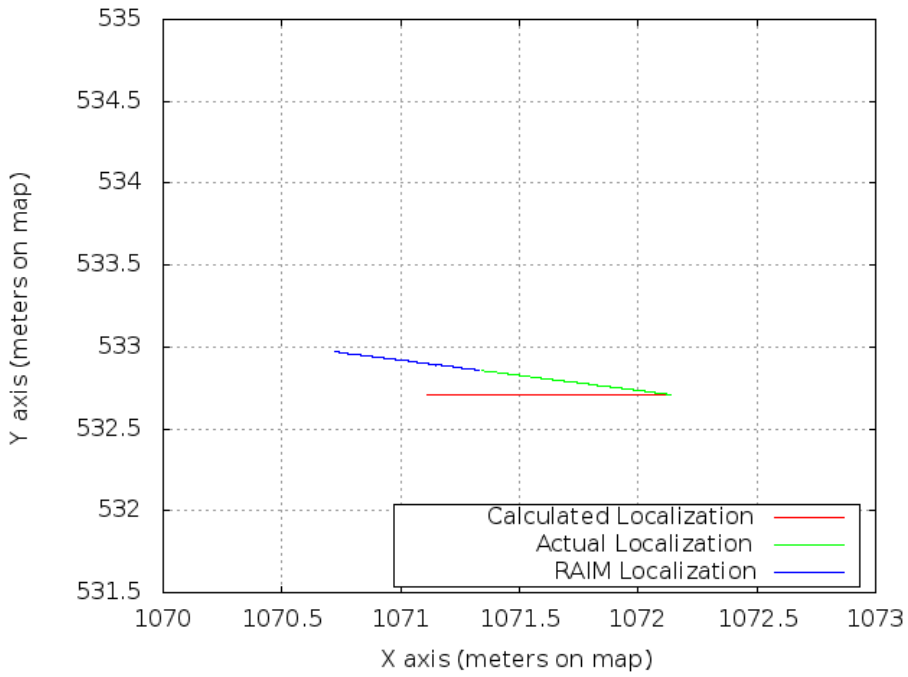

Figure 4-40: Y-value discrepancy detection and correction for RAIM duration=120s

ulation capability of high computing resource consumption attacks (from the testbed perspective) proves the UAVSim capability of simulating other complex attacks. These complex attacks will consume fewer resources of the underlying computing infrastructure.

- An increase in the number of attack hosts results in an increase of the average data loss and packet round trip times for all the simulations. Some irregularities were also found which could be caused by random placement of attack hosts when the number of attack hosts was changed in similar simulations.

- The number of attack hosts required to launch a jamming attack varied under different simulation conditions but, in general, a number greater than 30 was enough to launch a successful attack. For more sophisticated attacks, such as GPS Spoofing, only a single attack host was enough.

- With the increase in the number of attack hosts, the average packet loss increases linearly and the round trip time increases exponentially for both availability attacks.

- The average loss quickly became close to 100% for simulations of 10s while it decreased for longer simulations of 300s. This shows that during an attack, the system could not work for the first few minutes and may take some time to start communicating again. This also demonstrates the time it takes for the network to recover and stabilize in case of an attack using existing congestion control algorithms.

- On similar lines, it was observed that UAVs were not able to communicate with each other below a certain transmission power and at certain levels of transmission power, the loss was almost 50% without any attack in progress. Further, with an increased transmission power, even low power continuous jamming attacks were unsuccessful.

- The loss for all mobility models reach above 80% for as less as 10 attack hosts. Linear mobility model resembles the actual UAV path and has the lowest loss rate, which shows that if the location of UAV is unknown to attack hosts, it would require a large number of attack hosts to launch the attack successfully.

- Even for the Linear mobility model, the actual data loss (without any recovery measure in place) was about 20% initially and increased to 86% when the number of attack hosts is increased to 14 and becomes close to 99% when it is increased to 30. Also, this data is for a 5-minute network simulation with attack host packet transmission interval of 0.0001 seconds. For a 10-second simulation, the loss instantly reached 100%. This shows that if the UAV path is traceable and there is enough number of attack hosts in the vicinity, the system could become dysfunctional in as short as few seconds.

- Loss in a regular host that is not being attacked increases to about 60% when the UAVNet is under attack. This loss was earlier close to 20% for all communicating hosts irrespective of the number of UAV hosts in the network (which occurs due to limited transmission power). This shows that the increase in traffic in the area during the attack can also cause a mild jamming attack on other hosts that are not being directly targeted.

- The loss for the attacked host remains nearly 90% all the time irrespective of the number of UAV hosts in the network. This clearly indicates that an increased number of UAVs to ensure redundancy in the UAVNet is not of much help if some specific UAV hosts are targeted by an attacker. The attack will be successful and lead to total system failure in due time.

- Regarding GPS Jamming attack, the results were quite expected and similar to real-world jamming scenarios. As the number of attack hosts were increased, average GPS packet loss increased and reached up to 90% that indicates quite successful jamming. Clearly, UAV may go out of range of other UAVs and would render UAV unable to communicate with other UAVs.

- For GPS Spoofing attack, when discrepancy was introduced in only X-values, it was noticed that different motion paths have different variations, which implies that the variations could not be generalized.

- Discrepancy factor variation results in variation of the spoofed path as well. In case of circular path, this increase led to a spoofed linear path compared to a spoofed circular path when the factor was lower. Thus, low discrepancy factors would be hard to detect and can make a UAV lock on an attack host as a real satellite. The attack host can then increase the discrepancy factor to cause drastic path deviations.

- For GPS Spoofing attacks, it was noticed that generally, variation in Y-values are resulting in worse effects. In case of linear path, the resultant deviations were huge. In case of circular path, Y-value discrepancies caused resultant helical path, which could confuse the UAV and correction made to correct its path may lead to a crash.

- For GPS Spoofing attacks, the spoofed paths are generally similar to original paths in terms of the class of curve, i.e., spoofed paths for original linear paths were linear while for circular paths, they were curved paths. This would result in tougher detection of discrepancy or path deviation if the discrepancy factor is quite low.

- For RAIM implementation, it was noted that higher the discrepancy, easier and quicker it was to detect the error introduction. Further, a RAIM consistency check duration of 90 seconds worked best for various simulation experiments and was able to correct the path within few kilometers.

Chapter 5

Testbed Performance Evaluation

Focus of this chapter is to demonstrate the utility of UAVSim when used with generic computing infrastructure. In research or an academic setup, resources are constrained, and purchasing expensive high-end computing infrastructure becomes difficult. This limitation brings us back to one of the original requirements of the testbed - it should allow users to use it for any UAV network in a cost-effective manner. As mentioned in section 1.3, various in-house developed simulators that allow UAV swarm simulation, use high-end computing facilities and are proprietary. On the contrary, UAVSim has been designed to work with an existing simulation engine and other open source components. Thus, UAVSim is free to use and provides code editing at all levels of simulation. Simultaneously, UAVSim does not need expensive servers to run simulations but users might need to compromise on computation time.

Another important point to note is the increase in simulation run times for Jamming attack. DDoS attack works on the principle of sending a huge number of packets to one host causing network congestion and preventing it from communicating with other hosts. On the other hand, Jamming operates by transmitting pulses or noise signals on a broad frequency band to prevent communication in a wireless channel. Therefore, to implement a DDoS attack, a lesser number of attack hosts are required as only single host working at a frequency needs to be jammed. On the contrary,

Jamming requires all the frequencies to be jammed for a successful attack.

Most cyber-attacks that aim to take control over the subject involve transmission of particular command and control message instead of large amount of data transmission. Therefore, simulation of high computing resource consumption attacks (from the testbed perspective) would indicate the testbed's capability to simulate other lower resource consuming attacks.

5.1 System Setup

5.1.1 Hardware Setup

The PC we used for most simulations has an Intel CoreTM i7-3770 CPU (1 x 3.40 GHz 4-core, L2/L3 Cache: 1 MB/8 MB) and a system memory of 8.0 GB. The server used for server mode simulations has an Intel Xeon Processor E5-2630 (2 x 2.30 GHz 6-core, L2/L3 Cache: 1.5/15 MB) and a system memory of 64.0 GB. For concurrent multi-user simulations, the same PC was used to access the server using eight simultaneous terminal sessions.

5.1.2 Software Setup

Both of the systems defined in the previous paragraph, the PC and the server machines, run Ubuntu version 12.04 LTS. The server runs the 64-bit server version of Ubuntu (no GUI) while the PC is running the 64-bit desktop version with GUI. Both the machines have OMNeT++ version 4.2.2 with INET version 2.2 and CNI_OS3 version 1.0. As mentioned earlier, UAVSim uses OMNeT++ and these two open source modules to accurately simulate a UAVNet.

5.1.3 Simulation Setup

All simulations were 300 seconds long while the actual time taken to finish this simulation were observed. As mentioned earlier, the time it takes to attack highly secure time-sensitive military systems, such as a missile defense system, is only a few seconds. Even in our simulations, attacks were successful within few seconds of the start of the simulation. The basic abstracted UAV Model was used for our simulation as using more advanced models detailing various sub-modules would increase the simulation time about six folds. Advanced UAV models could also be used with the implementation of complex confidentiality or integrity compromising attacks. As an example, we did implement GPS Spoofing attack and for those simulations, we used the advanced UAV model with navigation. The frequency of UAV communication was fixed at 5 GHz for DDoS or single target attack scenario while it was varied between the range of 5- 15 GHz for Jamming or multiple target attack scenarios.

Several cases were simulated for different types of evaluation. For Case I, runtime and swarm behavior analysis, we varied both the number of attack hosts as well as the number of UAVs. In Case I_a, the number of UAVs was varied from 50 to 500 and in Case I_b, the number of attack hosts was varied from 2 to 20. In case of multiple frequency swarm simulations, the actual run time reached close to a day after 350 hosts, therefore, the simulations for this scenario in Case I_a was stopped at this number. Further, for checking the multi-user behavior, two separate analysis were done with separate cases. Case II, where the performance of swarm behavior with multiple concurrent users was evaluated. Case II_a, where 50 hosts were used for swarm behavior analysis while the concurrent number of users was increased from 1 to 8. Case II_b, the number of hosts, was increased to 100. Case III, where the performance of the testbed was evaluated for attack simulation. Case III_a involves use of five attack hosts and Case III_b uses ten attack hosts, keeping the number of

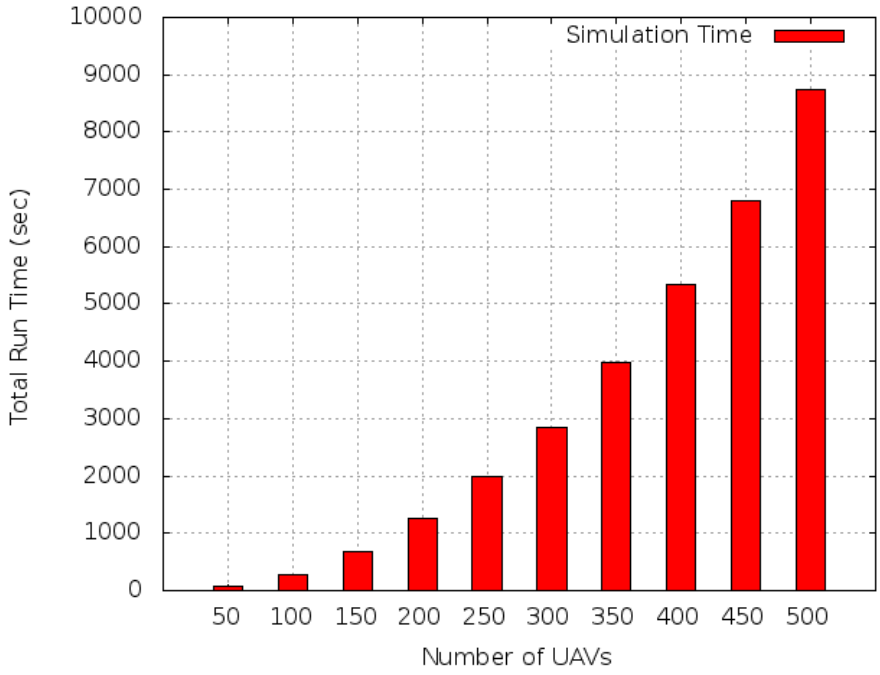

Figure 5-1: Run time variation with number of UAVs for single frequency swarm simulation

UAV hosts as 10 for both the scenarios. These three cases will be referred to during the discussion and analysis in the rest of the chapter.

5.2 DDoS

This subsection covers the results for all the simulations of DDoS attack. Various simulations were done for the above mentioned 3 cases, namely Case I, II and III. We have analyzed the effect of increasing number of UAVs, attack hosts, concurrent users in server mode operation and use of GUI.

5.2.1 Number of UAVs

For this analysis, we have used Case I_a (number of UAVs varied). As mentioned earlier, Case I_a involves the use of regular UAV hosts to determine the UAVSim capa-

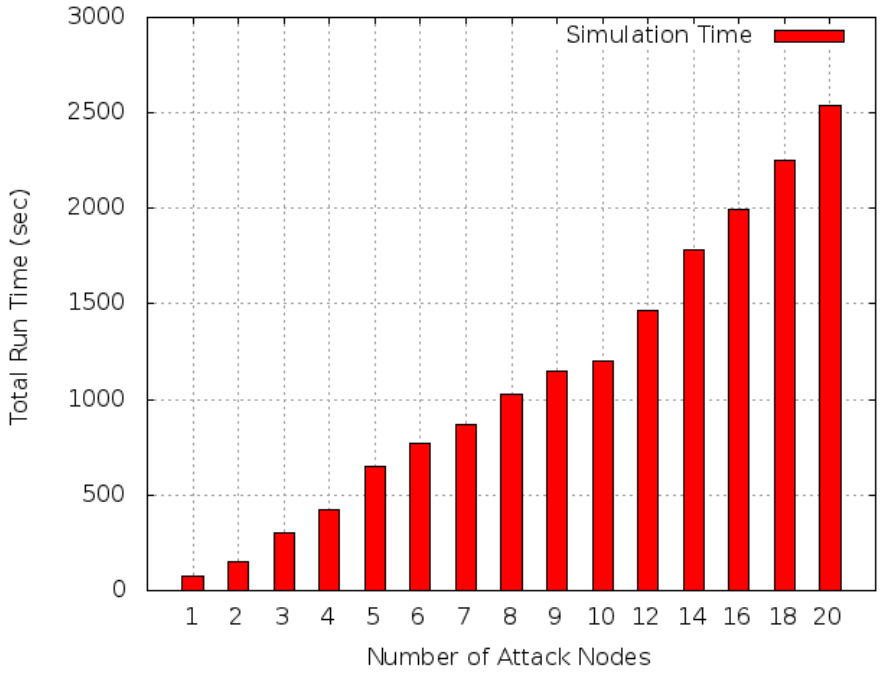

Figure 5-2: Run time variation with number of attack hosts for DDoS attack

bility of UAV swarm simulation, additional to the primary ability of UAVNet security simulation. It is clear from figure 5-1 that the testbed run-time varies exponentially with the increasing number of UAV hosts for this attack. Clearly, the run time is directly proportional to the powers of each 50 hosts and thus, can easily be predicted for a higher number of UAVs. Simulation time for 500 nodes is about an hour, and it indicates that a much larger number can be used for UAV swarm simulations while using a single frequency for communication.

5.2.2 Number of Attack Hosts

We used Case I_b (varying the number of attack hosts) for the second analysis. Figure 5-2 represents the performance for simulations with increasing number of attack hosts with the number of UAVs as 10. The trend clearly indicates that the runtime variation is exponential with respect to the number of attack hosts, with a multiple of 2 rather than 50 of the previous case. Therefore, a large number of attack hosts may

not be used for security simulations. Keeping in mind the number of attack hosts that can be simulated in a reasonable time, using a large value (more than 50) for this variable is not required.

5.2.3 Graphical User Interface

As a third analysis, we used the use of GUI (graphical user interface) which displays the network animation, as a performance metric. It is known that having a GUI to display the network behavior and statistics during a CPU intensive operation might impact the system performance. Therefore, we used Case I_b, where we varied the number of attack hosts and measured the speed of simulation for GUI and non-GUI options. Figure 5-3 show the results obtained for GUI and non-GUI options on the server as a green line and a red line, respectively. The blue line shows the percentage difference between the two modes with respect to the lower value (non-GUI option).

As shown in figure 5-3, the non-GUI runtime follows a non-linear polynomial trend with respect to the number of attack hosts. The percentage change between the GUI and non-GUI modes for a DDoS attack is not more than 7% for all cases with most cases being between $2 - 5\%$. Therefore, it can be said that the performance is not much affected by the use of GUI for this particular attack.

5.2.4 Number of Concurrent Users - Single Frequency Swarm Scenario

The fourth performance test was done by changing the number of simultaneous users accessing the simulation framework in parallel, using the high-speed server mode option. As discussed earlier, the server mode works only in a non-GUI mode to improve execution performance. We increased the number of parallel users from 1 to

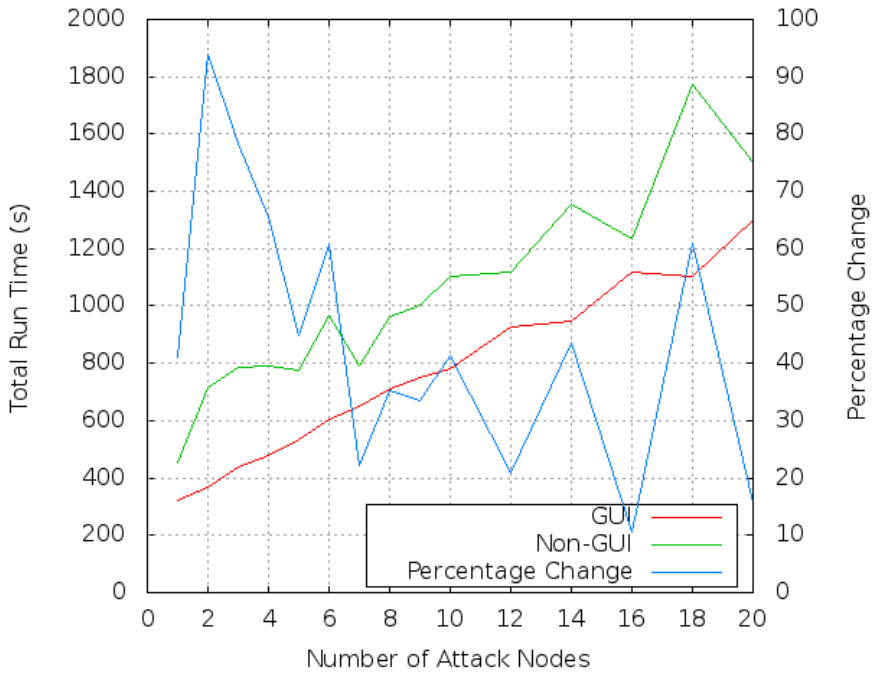

Figure 5-3: Run time variation with GUI and Non-GUI options, and the percentage change in two options for DDoS Attack

8 to check the effect on the performance of UAVSim in terms of average runtime for users.

Figure 5-4 show the evaluation results for Case II_a (50 UAV hosts) and II_b (100 UAV hosts) using the DDoS attack scenario. As mentioned earlier, in this scenario, there are no malicious hosts and all UAVs transmit at a frequency of 5 GHz. This simulation aims at analyzing the performance of UAVSim for multiple parallel users, simulating a swarm using single frequency in the absence of an attack. Please note that the two vertical axes show the variation of total run time for the two Cases, II_a and II_b. The error bars in the chart show the maximum and minimum time while the points depict the average time.

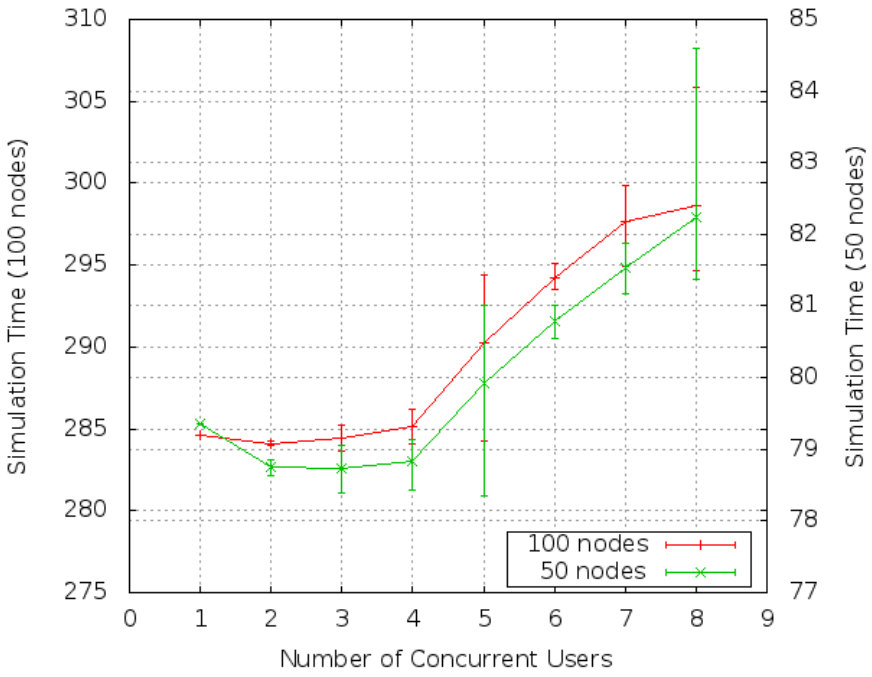

Figure 5-4: Run time variation with number of concurrent users in server mode operation for single frequency swarm simulation

5.2.5 Number of Concurrent Users - DDoS Security Simulation

The fifth performance metric for single frequency simulations involves the performance evaluation of UAVSim for DDoS attack simulations with multiple parallel users using it in server mode operation. To this end, Cases III_a (5 attack hosts) and III_b (10 attack hosts) were used. Figure 5-5 show the test results for this experiment. As mentioned earlier, the number of malicious hosts was changed for the two cases, keeping the number of UAV hosts as 10. The number of attack hosts used is much lesser than Case II because these attack hosts generate a lot more traffic in the network and cause an increase in the execution time. Similar to the previous analysis, the two vertical axes show the variation of total runtime for two different numbers of attack hosts. The error bars in the chart represent the maximum and minimum time

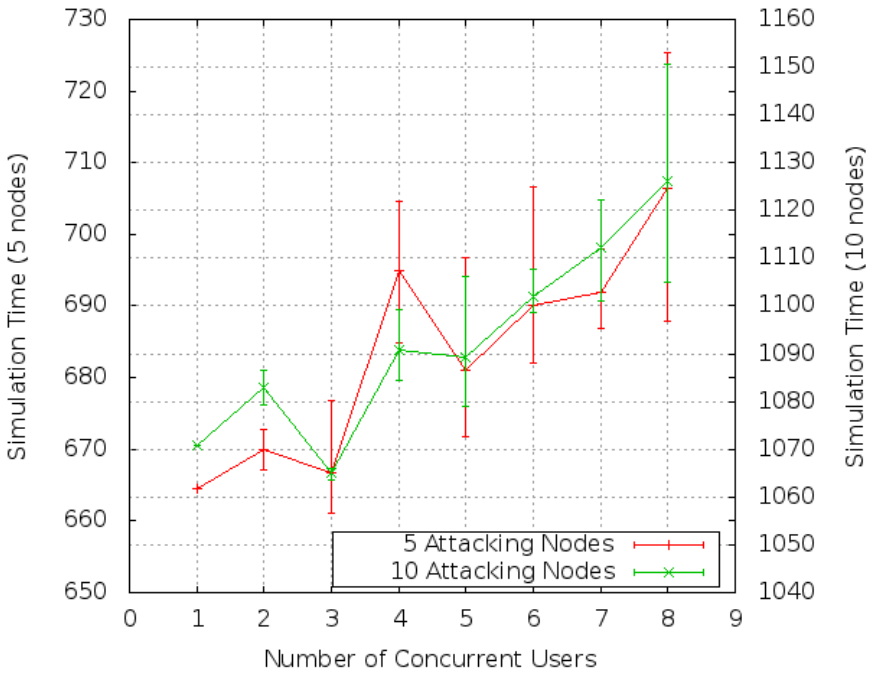

Figure 5-5: Run time variation with number of concurrent users in server mode operation for DDoS attack

while the points represent the average time.

5.3 Jamming Attack - Multiple Targets

This section covers the results of experiments for Jamming attack and multiple frequency simulation performances. These simulations were done for the 3 cases mentioned in subsection 5.1.3, namely Case I, II and III. We have analyzed the effect of increasing number of UAVs, attack hosts, concurrent users in server mode operation and use of GUI.

5.3.1 Number of UAVs

We have used Case I_a (number of UAVs varied) for this experiment. It should be noted that this simulation seems similar to DDoS attack UAV-only simulation,

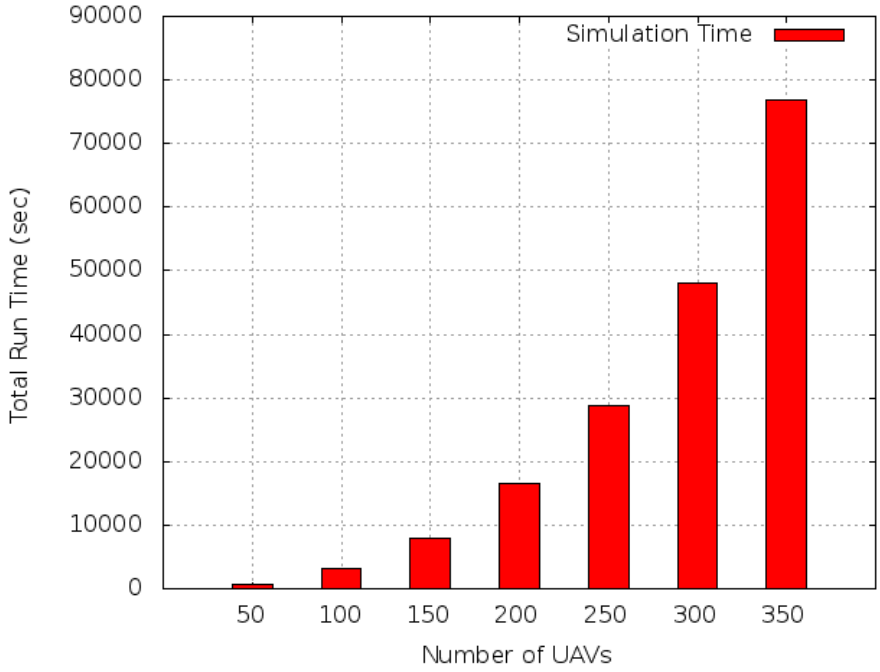

Figure 5-6: Run time variation with number of UAVs for multiple frequency swarm simulation

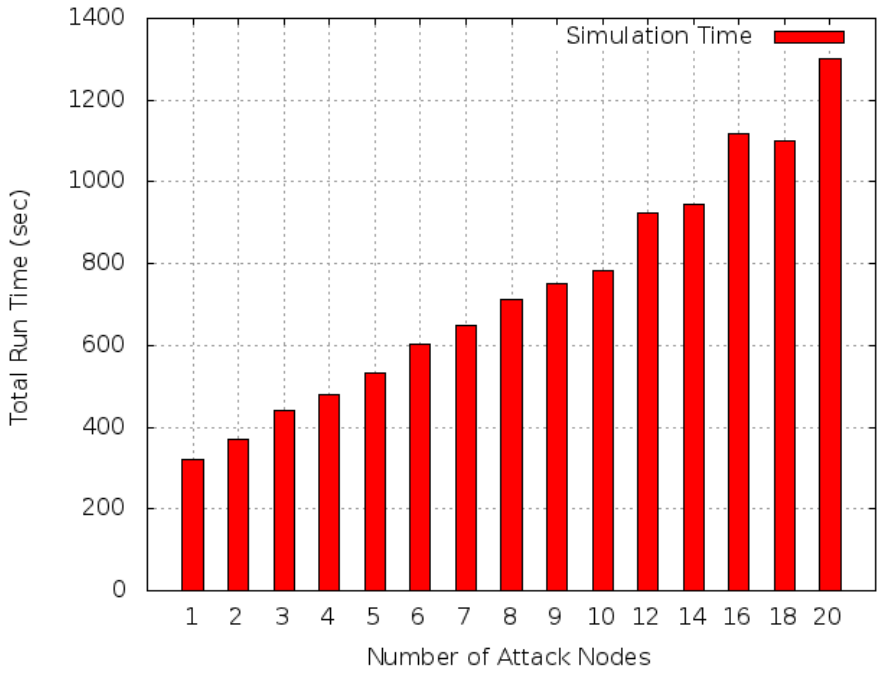

Figure 5-7: Run time variation with number of attack hosts for Jamming attack

but it differs in the use of multiple frequencies for communication rather than single. As mentioned earlier, the frequency range for UAV-UAV communication lies between 5-15 GHz for this case. This has been done to make sure that all frequencies are jammed in the attack area. It is clear from figure 5-6 that the testbed run-time varies exponentially with the increasing number of UAV hosts.

The simulation runtime, in this case, is directly proportional to the higher powers of each 50 hosts and thus, is easily predictable for a higher number of UAV hosts. Simulating a large number of UAVs would pose a challenge for this case because, for 350 nodes, the runtime reaches almost a day. It is also evident that the exponent might be higher than what was found in single frequency swarm simulation.

5.3.2 Number of Attack Hosts

Case I_b (varying the number of attack hosts) has been used for the second analysis of multiple frequency security simulation. Figure 5-7 shows the performance for simulations with increasing number of attack hosts with the number of UAV hosts fixed as 10. It is clear that the attack simulation trend for the multi-frequency attack simulation (Jamming attack) is non-linear polynomial instead of an exponential trend. Since the trend is not exponential, a large number of attack hosts may be used for security simulations of Jamming attack. From a comparison of the two attack scenarios, it is evident that the runtime for Jamming attack for Case I_b is roughly half of the DDoS attack simulation runtime. Once again, keeping in mind the number of attack hosts required for a successful attack, large numbers (more than 50) need not be used.

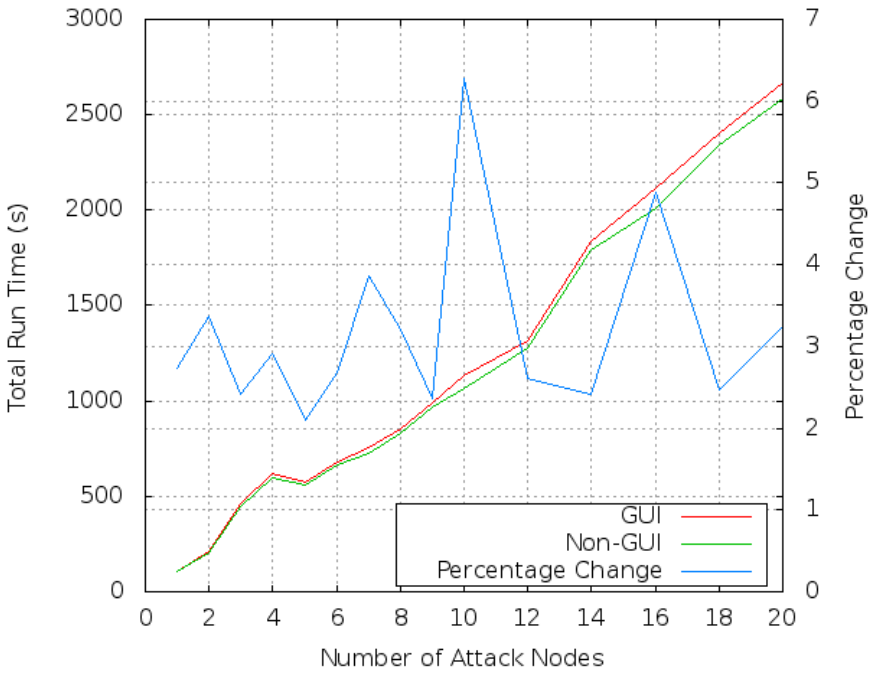

Figure 5-8: Run time variation with GUI and Non-GUI options, and the percentage change in two options for Jamming attack

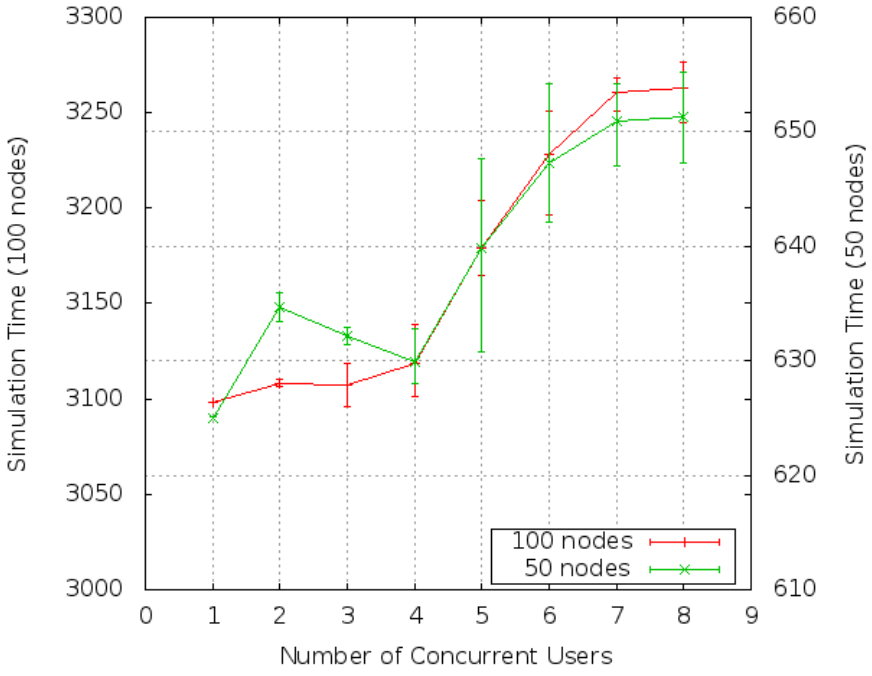

Figure 5-9: Run time variation with number of concurrent users in server mode operation for multiple frequency swarm simulation

5.3.3 Graphical User Interface

We used Case I_b (varying the number of attack hosts) to measure the speed of simulation for GUI and non-GUI options. Figure 5-8 present the results obtained for the GUI and non-GUI options in the server mode as a red line and a green line respectively, as indicated in the chart. The blue line (showing a heartbeat trend) represents the percentage variation between the two modes with respect to the lower value (non-GUI option).

The percentage variation between the GUI and non-GUI options for the Jamming attack is entirely random and higher for the lower number of attack hosts. Mostly, it is between $2 - 6\%$. This trend is completely opposite to the pattern obtained for DDoS attack simulation. It is evident that the performance is not affected by the use of GUI for a Jamming attack. The insignificant changes in performance for Jamming attack in current and previous case can be attributed to its already high simulation times.

5.3.4 Number of Concurrent Users - Multiple Frequency Swarm Simulation

This performance test was performed with a variable number of parallel users using the simulation testbed in the server mode option. The number of parallel users was increased from 1 to 8, and simulation run time for Cases II_a (50 UAV hosts) and II_b (100 UAV hosts) were evaluated. Figure 5-9 show the evaluation results using the multiple frequency swarm simulation scenario. These scenarios do not use any malicious host and all UAVs communicate at different frequencies in the range of 5-15 GHz. This simulation was meant to evaluate the performance for multiple parallel users, simulating a swarm with multiple frequencies in the absence of an attack. Please note that the two separate vertical axes show the variation of total run

time for the two Cases, II_a and II_b, lower numbers, of course, depicting Case II_a. The error bars in the chart show the maximum and minimum time while the points represent the average time.

It should be noted that both Cases II_a and II_b follow the similar trend after a certain number of users and the run time seems to be becoming invariable. Another important aspect to note is the average percentage variation in both cases is less than 5% between the time taken for single and 8 concurrent users.

5.3.5 Number of Concurrent Users - Jamming Attack Security Simulation

The final performance test for Jamming attack involves the performance analysis for the Jamming attack simulation with multiple users using UAVSim concurrently. Run time values for Cases III_a (5 attack hosts) and III_b (10 attack hosts) were evaluated. Figure 5-10 shows the performance test results for these cases. The two separate vertical axes in this evaluation also show the variation of total run time for two different numbers of attack hosts. The error bars in the chart represent the maximum and minimum time while the points represent the average time. It is evident that the plot for both cases follow similar trend once the number of concurrent users increases to three. Analogous to the previous evaluation of swarm simulation, the overall percentage variation between maximum and minimum run times for each case is less than 10%. The highest runtime variation for both instances was at 7 concurrent users while the lowest difference for Case III_a was at 2 and for Case III_b at 3 concurrent users.

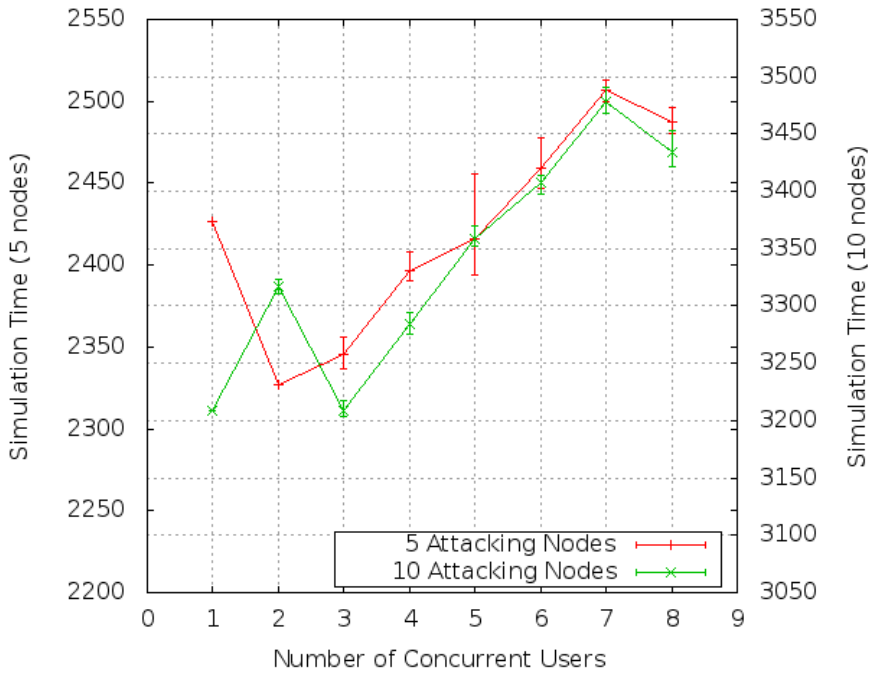

Figure 5-10: Run time variation with number of concurrent users in server mode operation for Jamming attack

5.4 Chapter Summary

In this chapter, several performance tests for variation in the number of UAVs, the number of attacks hosts, use of a GUI or non-GUI option and number of parallel users were performed. These tests gave us meaningful insights in terms of the estimated operational capacity of UAVSim. Although some simulation times were quite high in case of swarm simulations for a large number of hosts, the performance was very reasonable for security simulations. Some important points that can be noted from the analysis are as follows:

- Total run time varies exponentially with the number of attack hosts in security simulations as well as the number of UAVs in swarm simulation. It should be noted that despite the trend, the variation in these numbers is very different. The attack hosts change by 2 while, the UAVs change by 50. This gives a

clear indication of how many attack hosts and UAV hosts can be deployed in particular simulation scenario.

- Using the GUI for any security simulation has huge impact on performance for DDoS attacks, but the variations are very low for the Jamming attack. The Jamming attack requires more processing in terms of creation of channels for different frequencies and transmission of packets and results in slower execution times in general. Therefore, use of GUI doesn't impact these execution times any further.

- Performance evaluation for multiple parallel users using the testbed in server mode shows that performance is not greatly affected with increasing number of users, and average variation reduces as number of concurrent users increases more than four.

- The average simulation time saturates after a certain number of users and shows a trend of becoming invariable with respect to the number of concurrent users. The variation in minimum and maximum shows that total system performance is not affected much.

- It should be noticed that the simulation times for multiple parallel user analysis follows the same trend for a different number of UAVs or attack hosts once the number of users is more than four. This implies that irrespective of the number, the trend would be similar and thus, the runtime can easily be determined for higher number of users.

- The simulation run time for 20 attack hosts for both types of attacks took approximately half an hour. Practically, the number of attack host to launch such attacks is much less. For example, we need only one attack host for a successful GPS spoofing attack [104] and thus, the simulation capability is quite

extensive.

- The variation in run time for multiple parallel users not being exponential indicates the UAVSim's capability of handling more than 20 concurrent users. This particular evaluation was done for up to 8 users and showed an expected trend.

- Most of the trends follow the same pattern for both availability attacks. The run times are in general much higher for Jamming than the DDoS simulation. The main reason behind this is the anatomy of the simulation for these two attacks. Successful execution of a single target DDoS attack requires one host to be made unavailable while Jamming attempts to block a wide range of frequency, i.e., all hosts in the area be made unavailable. The underlying simulation engine of OMNeT++ simulates single object for a single channel (single frequency), and packets are transmitted on that channel within the same object (channel). Using several channels increases the inter-process communication between objects for packet transmission and causes an increase in total run time of simulations.

- Parts of results of this performance evaluation stage were also accepted as a conference paper [105] and a full article in a reputed peer-reviewed journal [106].

Chapter 6

Conclusion and Future Work

This chapter concludes this dissertation with a summary of achievements of this work and further discussion about possible future work directions in this area.

6.1 Conclusion

In essence, this work presents three significant contributions in the field of UAV security research - threat modeling and risk analysis of a UAV system; development of a simulation testbed environment for UAV networks; and demonstration of capability and usability of this testbed to simulate various attacks and their mitigation measures. To this end, several tasks were accomplished, and they are summarized below.

Although threat analysis of the Radio Communication of a UAV was already done [63], the overall security threat analysis of a UAV system was not accomplished until this work. The detailed threat analysis of the system is aimed at system architects as well as the users of the UAV system to help them be aware of, identify vulnerabilities and put mitigation and recovery measures for them. Since most of the information regarding measures already in place is confidential, it is still hard to identify which threats might affect the UAV systems most. What is certain is the presence of vulnerabilities in the UAV system, and the fact that many remain to be found. Adversaries are working day and night to attack our systems and any single

small vulnerability can't be ignored, to ensure the security of our systems. The development of a detailed threat model was the first achievement of this work and was published as a peer-reviewed article [15].

Further, we presented the UAVSim testbed, developed for security experiments of UAV networks in a cost-effective manner. UAVSim allows users to use different UAV models, simulate attacks and generate graphical results. Various simulations were performed to analyze the impact of two categories of the attack on the UAVS. These categories were Availability and Integrity attacks. Under the Availability attack class, we implemented the DDoS and Jamming attack against the UAV network. Navigation related attacks (GPS Spoofing and Jamming) were chosen for implementation under the integrity property violating attack class. Results from the analysis of these attack demonstrated the capability of the testbed to simulate the UAVNet as well various possible attacks. It was also shown that the testbed can be used for evaluating attacks and various mitigation measures. From a research perspective, UAVSim is a novel attempt to simulate the communication behavior of a UAVNet and the impact of attacks on the communication channels of this network rather than focusing on simulation of a single UAV. Development of this testbed is the second achievement of this work and its source code as well as detailed documentation (including user manual and installation manual) are available online.

Finally, the simulation run time analysis for high-resource consumption attacks as well as swarm simulations in UAVSim was presented to demonstrate its use in a generic computing environment. Various simulation results indicate that the performance of UAVSim is consistent, and it allows users to customize various options, according to their requirements. Along with attack simulation targeting single and multiple hosts, UAVSim was proved to be capable of simulating large UAV swarms. Performance for parallel multi-user operation in server mode was also evaluated for different scenarios. Although a scenario with a maximum of 8 concurrent users was

tested, it was established that for a reasonable number of simultaneous users, UAVSim will perform sufficiently fast. Interactive GUI, enhanced high-speed mode of operation, support of simultaneous users, etc. are some of the additional features which were discussed.

6.2 Future Work

As a future work, various mitigation techniques of currently available attacks in the attack library can be implemented. Although the impact of few attacks on UAV communication such as DDoS and Jamming have already been presented, more complicated attacks have yet to be implemented in this framework. The design of more UAV models for inclusion in the UAV models library as well as the design of more attacks for inclusion in the attack library also pose challenges and are potential future works. Further, new protocols, advanced attacks and various defensive and mitigation techniques could also be incorporated in UAVSim. Clearly, availability of real mission data will allow accurate testing of the design and other implementations.

References

[1] M. Kramer, "European Spacecraft Lands on Comet in Historic Space Feat." http://www.space.com/27740-rosetta-comet-landing-success.html, November 2014. Online; Last accessed: 3-June-2015.

[2] iRevolutions, "Humanitarian UAVs Fly in China After Earthquake (updated)." http://irevolution.net/2014/08/25/humanitarian-uav-china-earthquake/, August 2014. Online; Last accessed: 3-June-2015.

[3] B. Gabert, "NASA testing UAV to detect fires." http://fireaviation.com/tag/uav/, October 2014. Online; Last accessed: 3-June-2015.

[4] R. Austin, *UAV Design, Development and Deployment*. John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom: Wiley, fisrt ed., May 2010.

[5] K. Anderson and K. Gaston, "Lightweight unmanned aerial vehicles will revolutionize spatial ecology," *Frontiers in Ecology and the Environment*, vol. 11, pp. 138–146, March 2013.

[6] B. Handwerk, "5 Surprising Drone Uses (Besides Amazon Delivery)." http://news.nationalgeographic.com/news/2013/12/131202-drone-uav-uas-amazon-octocopter-bezos-science-aircraft-unmanned-robot/, December 2013. Online; Last accessed: 3-June-2015.

[7] H. Kelly, "Drones: The future of disaster response." http://whatsnext.

blogs.cnn.com/2013/05/23/drones-the-future-of-disaster-response/, May 2013. Online; Last accessed: 3-June-2015.

[8] D. Gilman, "Unmanned Aerial Vehicles in Humanitarian Response," June 2014. Online; Last accessed: 3-June-2015.

[9] F. Remondino, L. Barazzetti, F. Nex, M. Scaioni, and D. Sarazzi, "UAV Photogrammetry for Mapping and 3D Modeling - Curernt Status and Future Perspectives," in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (M. K. H. Eisenbeiss and H. Ingensand, eds.), (Zurich, Switzerland), Proceedings of the International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g), September 2011.

[10] G. Green, "Rhino-saving drones: How UAVs are being used for wildlife conservation." http://metro.co.uk/2014/05/09/rhino-saving-drones-how-uavs-are-being-used-for-wildlife-conservation-4721692/, May 2014. Online; Last accessed: 3-June-2015.

[11] Market Research Media, "U.S. Military Unmanned Aerial Vehicles (UAV) Market Forecast 2013-2018." http://www.marketresearchmedia.com/?p=509, January 2014. Online; Last accessed: 3-June-2015.

[12] FAA, *Unmanned Aircraft Systems*, November 2014. Online; Last accessed: 3-June-2015.

[13] K. Borgen, "Scratch build your own quad-copter!." http://www.instructables.com/id/Scratch-build-your-own-quad-copter/step2/Materials/, August 2013. Online; Last accessed: 3-June-2015.

[14] GPS World Staff, "Massive GPS Jamming Attack by North Korea." http://gpsworld.com/massive-gps-jamming-attack-by-north-korea/, May 2012. Online; Last accessed: 3-June-2015.

[15] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *2012 IEEE Conference on Technologies for Homeland Security (HST)*, pp. 585–590, November 2012.

[16] C. Whitlock, "Crashes mount as military flies more drones in U.S.." `http://www.washingtonpost.com/sf/investigative/2014/06/22/crashes-mount-as-military-flies-more-drones-in-u-s/`, June 2014. Online; Last accessed: 3-June-2015.

[17] S. Northcutt, "Are Satellites Vulnerable to Hackers?." `http://www.sans.edu/research/security-laboratory/article/satellite-dos`, May 2007. Online; Last accessed: 3-June-2015.

[18] W. Jayawardhana, "Intelsat to turn off LTTE beam." `http://archives.dailynews.lk/2007/04/13/news01.asp`, April 2007. Online; Last accessed: 3-June-2015.

[19] C. Arthur, "SkyGrabber: the $26 software used by insurgents to hack into US drones." `http://www.guardian.co.uk/technology/2009/dec/17/skygrabber-software-drones-hacked`, December 2009. Online; Last accessed: 3-June-2015.

[20] P. Meunier, "Drone Flaw Known Since 1990s." `http://www.cerias.purdue.edu/site/blog/post/drone_flaw_known_since_1990s_was_a_vulnerability/`, December 2009. Online; Last accessed: 3-June-2015.

[21] Associated Press Guardian, "Computer virus infects drone plane command center in US." `http://www.guardian.co.uk/technology/2011/oct/09/virus-infects-drone-plane-command`, October 2011. Online; Last accessed: 3-June-2015.

[22] F. Gardner, "Why Iran's capture of US drone will shake CIA." http://www.bbc.com/news/world-us-canada-16095823, December 2011. Online; Last accessed: 3-June-2015.

[23] B. Lendon, "Iran says it built copy of captured U.S. drone." http://www.cnn.com/2014/05/12/world/meast/iran-u-s-drone-copy/, May 2014. Online; Last accessed: 3-June-2015.

[24] M. Balali and M. Moghtader, "Iran says it shoots down Israeli spy drone." http://www.reuters.com/article/2014/08/24/us-iran-israel-idUSKBN0GO0II20140824, August 2014. Online; Last accessed: 3-June-2015.

[25] B. J. Carlson, "Past UAV Program Failures and Implications for Current UAV Programs," tech. rep., DTIC Document, 2001.

[26] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad hoc networks*, vol. 2, no. 4, pp. 351–367, 2004.

[27] T. Pappaport, A. Annamalai, R. Buehrer, *et al.*, "Wireless communications: past events and a future perspectire," *IEEE Communications Magzine*, vol. 5, pp. 148–161, 2002.

[28] K. M. Loong and G. Leng, "Sensors and Communications Range Relations for UAV Operations," in *New Challenges in Aerospace and Technology Maintenance Conference, Singapore*, 2006.

[29] J. Li, Y. Zhou, and L. Lamont, "Communication architectures and protocols for networking unmanned aerial vehicles," in *Globecom Workshops (GC Wkshps), 2013 IEEE*, pp. 1415–1420, December 2013.

[30] P. Lu and Q. Geng, "Real-time simulation system for UAV based on Mat-

lab/Simulink," in *2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering (CCIE)*, vol. 1, pp. 399–404, August 2011.

[31] J. Zhang, Q. Geng, and Q. Fei, "UAV flight control system modeling and simulation based on FlightGear," in *International Conference on Automatic Control and Artificial Intelligence (ACAI 2012)*, pp. 2231–2234, March 2012.

[32] A. Kim, B. Wampler, J. Goppert, I. Hwang, and H. Aldridge, "Cyber attack vulnerabilities analysis for unmanned aerial vehicles," *The American Institute of Aeronautics and Astronautics: Reston, VA, USA*, 2012.

[33] J. Goppert, A. Shull, N. Sathyamoorthy, W. Liu, I. Hwang, and H. Aldridge, "Software/Hardware-in-the-Loop Analysis of Cyberattacks on Unmanned Aerial Systems," *Journal of Aerospace Information Systems*, vol. 11, no. 5, pp. 337–343, 2014.

[34] Y. Qiang, X. Bin, Z. Yao, Y. Yanping, L. Haotao, and Z. Wei, "Visual simulation system for quadrotor unmanned aerial vehicles," in *2011 30th Chinese Control Conference (CCC)*, pp. 454–459, July 2011.

[35] T. X. Brown and S. K. Doshi and S. Jadhav and J. Himmelstein, "Test Bed for a Wireless Network on Small UAVs," in *In Proceedings of AIAA 3rd Unmanned Unlimited Technical Conference*, pp. 20–23, 2004.

[36] J. Wu, W. Wang, J. Zhang, and B. Wang, "Research of a kind of new UAV training simulator based on equipment simulation," in *2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)*, vol. 9, pp. 4812–4815, August 2011.

[37] J. Yang and H. Li, "UAV Hardware-in-loop Simulation System Based on Right-angle Robot," in *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 1, pp. 58–61, August 2012.

[38] J. Corner and G. Lamont, "Parallel simulation of UAV swarm scenarios," in *Proceedings of the 2004 Winter Simulation Conference*, vol. 1, pp. –363, December 2004.

[39] S. Hamilton, T. Schmoyer, and J. D. Hamilton, "Validating a network simulation testbed for army UAVs," in *2007 Winter Simulation Conference*, pp. 1300–1305, December 2007.

[40] Metron High Perfomance Computing, "SPEEDES." `http://www.speedes.com/`. Online; Last accessed: 3-June-2015.

[41] M. Ashtiani and M. A. Azgomi, "A distributed simulation framework for modeling cyber attacks and the evaluation of security measures," *Simulation*, p. 0037549714540221, 2014.

[42] E. Pereira, K. Hedrick, and R. Sengupta, "The C3UV Testbed for Collaborative Control and Information Acquisition Using UAVs," in *2013 American Control Conference (ACC)*, pp. 1466–1471, IEEE, 2013.

[43] S. Chaumette, R. Laplace, C. Mazel, and R. Mirault, "SCUAL, swarm of communicating UAVs at LaBRI: An open UAVNet testbed," in *Wireless Personal Multimedia Communications (WPMC), 2011 14th International Symposium on*, pp. 1–5, IEEE, 2011.

[44] FlightGear Wiki, "Global Positioning System." `http://wiki.flightgear.org/Global_Positioning_System`, December 2013. Online; Last accessed: 3-June-2015.

[45] Jarontec, "UAVPlayground An approach to Unmanned Aerial Vehicles (UAV) with Java and the Processing Development Environment." `https://code.google.com/p/uavplayground/`, August 2010. Online; Last accessed: 3-June-2015.

[46] IDS Corporation Aeronautical Division, "Hero UAVSim Unmanned Aerial Vehicle Simulator." https://www.idscorporation.com/images/aeronautical/homepage/BRO_AERO_HEROSIM.pdf, September 2013. Online; Last accessed: 3-June-2015.

[47] IDS Corporation Aeronautical Division, "Hero UAVSim Unmanned Aerial Vehicle Simulator." https://www.idscorporation.com/images/aeronautical/homepage/BRO_AERO_HEROGCS.pdf, September 2013. Online; Last accessed: 3-June-2015.

[48] IDS Corporation Aeronautical Division, "Hero UAVSim Unmanned Aerial Vehicle Simulator." https://www.idscorporation.com/images/aeronautical/homepage/BRO_AERO_HEROUAV.pdf, September 2013. Online; Last accessed: 3-June-2015.

[49] LabSat, "LabSat GPS Simulator." http://www.labsat.co.uk/index.php/en/. Online; Last accessed: 3-June-2015.

[50] Spirent, "GPS Simulation Simulate GPS signals for professional, controllable and testing in the lab." http://www.spirent.com/Positioning-and-Navigation/GPS_Simulation. Online; Last accessed: 3-June-2015.

[51] National Instruments, "NI GPS Simulator PXI RF Test System for GPS Receiver Test." http://sine.ni.com/nips/cds/view/p/lang/en/nid/206805. Online; Last accessed: 3-June-2015.

[52] F. Zimmermann, T. Haak, and C. Hill, "Galileo system simulation facility," in *8th International Workshop on Simulation for European Space Programmes, SESP*, 2004.

[53] F. Schubert, R. Prieto-Cerdeira, P. Robertson, and B. H. Fleury, "SNACS–

The Satellite Navigation Radio Channel Signal Simulator," *Proc ION GNSS. Institute of Navigation, Savannah, GA*, pp. 1982–1988, 2009.

[54] A. J. Turner, *An open-source, extensible spacecraft simulation and modeling environment framework*. PhD thesis, Virginia Polytechnic Institute and State University, 2003.

[55] A. Y. Javaid, W. Sun, and M. Alam, "UAVSim: A simulation testbed for unmanned aerial vehicle network cyber security analysis," in *2013 IEEE Globecom Workshops (GC Wkshps)*, pp. 1432–1436, December 2013.

[56] E. M. Puchaty and D. A. DeLaurentis, "A performance study of UAV-based sensor networks under cyber attack," in *System of Systems Engineering (SoSE), 2011 6th International Conference on*, pp. 214–219, IEEE, 2011.

[57] Z. Goraj, "UAV platforms designed in WUT for border surveillance," *structure (total)*, vol. 14, p. 8, 2007.

[58] P. Katopodis, G. Katsis, O. Walker, M. Tummala, and J. Michael, "A Hybrid, Large-scale Wireless Sensor Network for Missile Defense," in *IEEE International Conference on System of Systems Engineering, 2007. SoSE '07.*, pp. 1–5, April 2007.

[59] AeroVironment, "AeroVironment Develops Worlds First Fully Operational Life-Size Hummingbird-Like Unmanned Aircraft for DARPA." `http://www.avinc.com/resources/press_release/aerovironment_develops_worlds_first_fully_operational_life-size_hummingbird`, February 2011. Online; Last accessed: 3-June-2015.

[60] T. Koo, D. Shim, O. Shakernia, B. Sinopoli, Y. Ma, F. Hoffmann, and S. Sastry, "Hierarchical hybrid system design on Berkeley UAV," *International Aerial Robotics Competition*, 1998.

[61] E. W. Justh and P. S. Krishnaprasad, "A simple control law for UAV formation flying," tech. rep., DTIC Document, 2002.

[62] J. Evans, G. Inalhan, J. S. Jang, R. Teo, and C. J. Tomlin, "Dragonfly: A versatile UAV platform for the advancement of aircraft navigation and control," in *Digital Avionics Systems, 2001. DASC. 20th Conference*, vol. 1, pp. 1C3–1, IEEE, 2001.

[63] D. Rudinskas, Z. Goraj, and J. Stankūnas, "Security analysis of UAV radio communication system," *Aviation*, vol. 13, no. 4, pp. 116–121, 2009.

[64] T. Reed, J. Geis, and S. Dietrich, "SkyNET: A 3G-Enabled Mobile Attack Drone and Stealth Botmaster.," in *WOOT*, pp. 28–36, 2011.

[65] F. Anjum and P. Mouchtaris, *Security for wireless ad hoc networks*. John Wiley & Sons, 2007.

[66] S. Dredge, "Lily: the $499 selfie drone that's your personal videographer." `http://www.theguardian.com/technology/2015/may/13/lily-drone-personal-videographer-selfie`, May 2015. Online; Last accessed: 3-June-2015.

[67] R. Crook, D. Ince, L. Lin, and B. Nuseibeh, "Security requirements engineering: when anti-requirements hit the fan," in *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pp. 203–205, 2002.

[68] E. Oladimeji, S. Supakkul, and L. Chung, "Security threat modeling and analysis: A goal-oriented approach," in *Proc. of the 10th IASTED International Conference on Software Engineering and Applications (SEA 2006)*, pp. 13–15, Citeseer, 2006.

[69] D. LeBlanc and M. Howard, *Writing secure code*. Pearson Education, 2002.

[70] F. Swiderski and W. Snyder, *Threat modeling*. Microsoft Press, 2004.

[71] H. H. Thompson, "Application penetration testing," *IEEE Security & Privacy*, vol. 3, no. 1, pp. 66–69, 2005.

[72] W. Wang, Y. Sun, H. Li, and Z. Han, "Cross-layer attack and defense in cognitive radio networks," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pp. 1–6, December 2010.

[73] J. Alves-Foss, "Multi-protocol attacks and the public key infrastructure," in *Proc. 21st National Information Systems Security Conference*, pp. 566–576, 1998.

[74] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," *Proceedings of the National Academy of Sciences*, vol. 108, no. 10, pp. 3838–3841, 2011.

[75] R. Poisel, *Modern Communications Jamming: Principles and Techniques*. Artech House, 2011.

[76] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 245–257, 2011.

[77] G. Carle, F. Dressler, R. A. Kemmerer, H. Koenig, C. Kruegel, and P. Laskov, "Network attack detection and defense," in *Manifesto of the Dagstuhl Perspectives Workshop, March*, pp. 2–6, 2008.

[78] M. Barbeau, "Wimax/802.16 threat analysis," in *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, pp. 8–15, ACM, 2005.

[79] G. Loukas, D. Gan, and T. Vuong, "A review of cyber threats and defence approaches in emergency management," *Future Internet*, vol. 5, no. 2, pp. 205–236, 2013.

[80] M. S. Faughnan, B. J. Hourican, G. C. MacDonald, M. Srivastava, J. Wright, Y. Haimes, E. Andrijcic, Z. Guo, and J. White, "Risk analysis of unmanned aerial vehicle hijacking and methods of its detection," in *Systems and Information Engineering Design Symposium (SIEDS), 2013 IEEE*, pp. 145–150, IEEE, 2013.

[81] A. R. Perry, "The flightgear flight simulator," in *Proceedings of the USENIX Annual Technical Conference*, 2004.

[82] ETSI Technical Specification, "102 165-1: Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4, V4.1.1," *Interoperability test methods and approaches*, 2003.

[83] H. Lv, "Research on network risk assessment based on attack probability," in *Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on*, vol. 2, pp. 376–381, IEEE, 2009.

[84] S. Mehta, K. Kwak, and N. Sulatan, *Network and system simulation tools for next generation networks: a case study*. INTECH Open Access Publisher, 2010.

[85] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM-2001)*, vol. 9, p. 185, sn, 2001.

[86] B. Niehoefer, S. Šubik, and C. Wietfeld, "The CNI Open Source Satellite Simulator Based on OMNeT++," in *Proceedings of the 6th International ICST*

Conference on Simulation Tools and Techniques, SimuTools '13, (ICST, Brussels, Belgium, Belgium), pp. 314–321, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.

[87] I. C. Wietfeld, "The Open Source Satellite Simulator." `http://www-os3.kn.e-technik.tu-dortmund.de/index.php`, 2014. Online; Last accessed: 3-June-2015.

[88] A. Lewandowski, R. Burda, and C. Wietfeld, "A Multiscale Real-time Navigation and Communication Satellite Simulation Model for OMNeT++," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, (ICST, Brussels, Belgium, Belgium), pp. 87:1–87:8, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[89] A. Varga *et al.*, "INET framework for OMNeT++ 4.0." `http://inet.omnetpp.org/`, May 2009. Online; Last accessed: 3-June-2015.

[90] A. Amato, "Is GPS Connectivity Important for Your Drone." `http://dronelife.com/2014/10/24/drone-need-gps-connection/`, October 2014. Online; Last accessed: 3-June-2015.

[91] B. Boughton, "Unmanned Aerial Vehicles (UAV) in Precision Agriculture." `http://agmapsonline.com/?p=624`, January 2014. Online; Last accessed: 3-June-2015.

[92] M. Thomas *et al.*, "Global Navigation Space Systems: reliance and vulnerabilities," 2011.

[93] A. Rhattoy and A. Zatni, "The impact of radio propagation models on ad hoc

networks performances," *Journal of Computer Science*, vol. 8, no. 5, p. 752, 2012.

[94] P. Zhan, K. Yu, *et al.*, "Wireless relay communications with unmanned aerial vehicles: Performance and optimization," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 47, no. 3, pp. 2068–2085, 2011.

[95] K. M. Elleithy, D. Blagovic, W. Cheng, and P. Sideleau, "Denial of Service Attack Techniques: Analysis, Implementation and Comparison," *Journal of Systemics, Cybernetics and Informatics*, vol. 3, pp. 66–71, 2006.

[96] T. Karygiannis and L. Owens, "Wireless network security," *NIST special publication*, vol. 800, p. 48, 2002.

[97] D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Drone Hack: Spoofing Attack Demonstration on a Civilian Unmanned Aerial Vehicle." `http://gpsworld.com/drone-hack/`, August 2012. Online; Last accessed: 3-June-2015.

[98] D. P. Shepard, J. A. Bhatti, T. E. Humphreys, and A. A. Fansler, "Evaluation of smart grid and civilian UAV vulnerability to GPS spoofing attacks," in *Proceedings of the ION GNSS Meeting*, 2012.

[99] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, and P. M. K. Jr., "Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008),Savannah, GA, September 2008, pp. 2314-2325.*, 2008.

[100] B. W. O'Hanlon, M. L. Psiaki, T. E. Humphreys, and J. A. Bhatti, "Real-time spoofing detection in a narrow-band civil GPS receiver," *Proc. ION GNSS 2010*, pp. 21–24, 2010.

[101] N. Kim, "Interference effects on gps receivers in weak signal environments," *University of Calgary, Department of Geomatics Engineering Masters Thesis*, 2006.

[102] S. Hewitson and J. Wang, "GNSS receiver autonomous integrity monitoring with a dynamic model," *Journal of Navigation*, vol. 60, no. 02, pp. 247–263, 2007.

[103] ESA NAVipedia, "RAIM." `http://www.navipedia.net/index.php?title=RAIM&oldid=13305`, September 2014. Online; Last accessed :3-June-2015.

[104] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the Requirements for Successful GPS Spoofing Attacks," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, (New York, NY, USA), pp. 75–86, ACM, 2011.

[105] A. Y. Javaid, W. Sun, and M. Alam, "Single and Multiple UAV Cyber-Attack Simulation and Performance Evaluation," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 1, 2014.

[106] A. Y. Javaid, W. Sun, and M. Alam, "UAVNet Simulation in UAVSim: A Performance Evaluation and Enhancement," in *9th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM 2014)*, May 2014.

Appendix A

Online Resources

Several project resources including the source code are available online, and their corresponding URLs have been provided here for future reference:

1. Project Page: `http://www.yazdan.us/research`
2. Source Code: `https://github.com/ayjavaid/OMNET_OS3_UAVSim`
3. Installation Manual: `http://www.yazdan.us/InstallationManualv1.0.pdf`
4. User Manual: `http://www.yazdan.us/UserManualv1.0.pdf`