Design and Implement Deep-Space network and validate its existence against De-authentication attack

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science and Engineering

Submitted By
Kritika Sharma
(801732027)

Under the supervision of:
Dr. Maninder Singh
Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA – 147004

July 2019

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Design and Implement Deep-Space network and validate its existence against De-authentication attack*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Maninder Singh* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:

(Kritika Sharma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Maninder Singh)
Professor and Head,
CSED

i

ACKNOWLEDGEMENT

First of all, I would like to thank the Almighty, who has always guided me to walk on the right path of life. It is a great privilege to express my gratitude towards my respected supervisor **Dr. Maninder Singh,** Professor of Computer Science and Engineering Department. He has been an esteemed guide and great support behind achieving this task. This work would not have been possible without the encouragement and able guidance of him. I thank my supervisor for his time, patience, discussions and valuable comments. His enthusiasm and optimism made this experience both rewarding and enjoyable. I am truly grateful to him for extending his co-operation and understanding whenever I need help and guidance from him. I am also heartily thankful to **Dr. Ashutosh Mishra**, P.G. coordinator, for motivation and providing uncanny guidance and support throughout the preparation of the thesis.

I will be failing in my duty if I do not express my gratitude to **Dr. S. S. Bhatia**, Senior Professor and Dean of Academic Affairs, for making provisions of infrastructure such as library facilities, computer labs with internet facilities, immensely useful for the learners to equip themselves with the best in the field.

I also want to express my sincere gratitude to **Mr. Mohit**, PhD student, for helping me and guiding me through this journey. I thank him for his perseverance and his patience. Without him this thesis would not be possible.

I am also thankful to the entire staff members of CSED for their direct or indirect help, co-operation, love, and affection, which made my stay at Thapar Institute of Engineering and Technology memorable. Last but not least, I would like to thank my family for their wonderful love and encouragement, without their blessings none of this would be possible.

Kritika Sharma

(801732027)

ABSTRACT

Deep Space is full of challenges when it comes to network communication that keeps motivating us to come up with a technology that can withstand constraints and challenges related to this communication. Some of the prominent challenges of Deep Space are intermittent connectivity, huge propagation delays, blackouts, high retransmission rates, high error rates, etc. Consider a scenario in which Google needs to be accessed on Mars. The communication starts by typing in the URL address of Google which afterward is translated to an IP address by DNS. A request for translation is sent to the Earth's Deep Space Stations (DSS) that is currently in communication with the satellite. In deep space, Round Trip Time (RTT) for communication depends on the current positions of planets for e.g. Earth and Mars communication last somewhere between 7-40 minutes depending on how they are positioned. Even the delays for Geosynchronous satellites last about 250ms. Say it took 14 mins for the request to be handled and in the meantime, the station with which the communication was being held could have moved as planets are constantly in motion i.e. revolving around the sun and rotating about their own axis. This might result in Earthly devices acquiring a different IP address [21]. As a result, communication is not feasible between the two devices.

To overcome such challenges researchers came up with Delay and Disruption Tolerant Networking that is independent of end to end path and can handle harsh environments of space. DTN will allow us to test the limits of how far we can go in deep space. Many researchers and various international organizations are working to make it operable to realize its capabilities.

But as the technology advances so do the threats associated with it. Space is difficult territory and does not have boundaries as such, which are prevalent in terrestrial networks. If something goes wrong, there isn't much to rely on. On Earth, even in an event of an attack, it can be resolved with not that much trouble but if this was to happen in Deep Space

where the RTT alone takes minutes to hours how can any prolific help be provided in such situation. Security is an important parameter to consider while communicating. As the networks in deep space are wireless, they are prone to a huge number of threats.

This research work deals with the query that whether the attacks done on terrestrial networks are also possible in Deep Space as they use the same underlying protocols so they might be prone to the same vulnerabilities. If the attacks are possible how adversely will they affect the productivity of networks? The attack performed for validating the doubts presented is De-authentication attack that disconnects the two DTN nodes communicating to each other in an environment like deep space. This thesis also explores the detection and prevention mechanism for the attacks carried out. The detection mechanism produces an output with the Basic Service Set Identifier (BSSID) of the target and the count of packets addressed to it. There are some mechanisms listed that can be used to protect the devices against the attack performed to make the communication infrastructure secure.

TABLE OF CONTENTS

TITLE	PAGE NO.
Certificate	i
Acknowledgment	ii
Abstract	iii
Table of Contents	v
List of Figures	vii
List of Tables	ix
List of Abbreviations	x
1. Introduction	1-4
1.1. Introduction to Delay Tolerant Networks (DTN)	2
1.2. Basic Idea and Outline of the Thesis	3
2. Literature Review	5-16
2.1. Deep Space Communication	5
2.2. Interplanetary Internet (IPN)	7
2.2.1 Architecture of IPN	7
2.3. Delay Tolerant Network (DTN)	9
2.4. Difference between DTN and Terrestrial Internet	10
2.5. Practical Implementations of DTN	12
2.5.1 United Kingdom-Disaster Monitoring Satellite (UK-DMC)	12
2.5.2 DTN implementation of Epoxi Spacecraft	13
2.5.3 DakNet	13
2.6. De-authentication Attack	14
2.7 Interplanetary Overlay Network (ION)	15
3. Research Gaps	17-18
3.1 Objectives	17
4. Design and Implementation of Deep Space Network	19-37

4.1. Testbed	27
4.1.1 Preparing the Testbed	31
4.2. Testing	31
4.2.1 Examining the network by sending a file	32
4.2.2 Launching De-authentication attack on the network	33
4.2.3 Detecting the De-authentication Attack	37
5. Results and Discussions	38-43
5.1. Test A: Time Based De-authentication attack	39
5.2. Test B: Contact window exhaustion attack	40
5.3. Protection Mechanisms	42
6. Conclusions and Future Scope	44
References	45-47
Appendices	48-55
A. Installing ION software	48
B. End to End transmission of data from spacecraft to MCC	50
C. Codes Used in Thesis	52
C.1 Code to find SSID, Channel Number and BSSID of an access point.	52
C.2 Code for De-authentication attack	54
C.3 Code for Detecting De-authentication attack	56
C.4 Code to find hidden SSID	57
C.5 Code to find Clients of specific Access Point	57

LIST OF FIGURES

Fig 1. Current status of the Internet on Earth 1

Fig 2. Comparison between TCP/IP and DTN protocol stack. 3

Fig 3. Map of Deep Space Stations 6

Fig 4. JPL's Deep Space Network Now that shows Currently active DSS [6] 7

Fig 5. Interplanetary Internet Architecture [30] 8

Fig 6. Planetary Network Architecture [5] 9

Fig 7. DTN Protocol Stack in communication 10

Fig 8. Packet traversal from Source to Destination in Terrestrial Internet [27] 11

Fig 9. Bundle traversal from Source to Destination in IPN [27]. 12

Fig 10. DTN protocol Stack 15

Fig 11. Graphical representation of Attack on Network 20

Fig 12. Flowchart of the work done 26

Fig 13. Physical setup of testbed 27

Fig 14. Logical setup of Testbed 28

Fig 15. Configuration file for two nodes communicating via LTP/UDP 30

Fig 16. Shellcode to send multiple files on DTN network 31

Fig 17. Loading configuration file for Node 2 into the system 32

Fig 18. Execution of shellcode to send a file from Node 1 32

Fig 19. An entry about receiving a file at Node 2 in ion.log file 33

Fig 20. Bundle captured in Wireshark while sending a file from Node 1 to Node 2 33

Fig 21. Execution of shellcode to send multiple files to Node 2 34

Fig 22. Entries in the ion.log file about receiving files from Node 1 34

Fig 23. Multiple bundles transmitted across the network as shown by Wireshark 34

Fig 24. Wireless adapter has been added to attacking machine 35

Fig 25. Changing mode of wireless card from Managed to Monitor in order to sniff traffic 35

Fig 26. Python code finding out BSSID, Channel and SSID of different Access Points 35

Fig 27. Attacking the nodes connected to DTN using python based De-authentication code 36

Fig 28. Node disconnects from the network 36

Fig 29. De-authentication packets sent to node captured in Wireshark 36

Fig 30. Output of code that detects the De-authentication attack 37

Fig 31. Bundle Transmission from Node 1 to Node 2 38

Fig 32. Bundle transmission in Test A: Time-based De-authentication attack 39

Fig 33. Comparison of transmission of Bundles when normal transmission is happening and when attack happens 40

Fig 34. Bundles transmission in Test B: Contact window exhaustion attack 41

Fig 35. Analysis of bundle delivery due to Attack 41

Fig 36. Commands for installing Expat XML parser 48

Fig 37. Commands for installing ION software 49

Fig 38. Data transmission from Spacecraft to Mission Control Center (MCC) 50

LIST OF TABLES

Table 1. Successors of UK-DMC 12

Table 2. The course of the DINET Experiment 13

Table 3. Configuration details of Nodes 27

Table 4 Protocols that are supported by ION 48

LIST OF ABBREVIATIONS

IPN	Interplanetary Internet
DTN	Delay Tolerant Network
DSN	Deep Space Network
NEN	Near Earth Network
SN	Space Network
DSS	Deep Space Station
MCC	Mission Control center
NASA	National Aeronautics and Space Administration
JPL	Jet Propulsion Labs
LEO	Low Earth Orbit
GEO	Geosynchronous Earth Orbit
CCSDS	Consultative Committee for Space Data System
BP	Bundle Protocol
LTP	Licklider Transport Protocol
CFDP	CCSDS File Delivery Protocol
RTT	Round Trip Time
UK-DMC	United Kingdom Disaster Management Satellite
DINET	Deep Impact Network Environment Training
IP	Internet Protocol
LTPCL	Licklider Transport Protocol Convergence Layer
SSID	Service Set Identifier
BSSID	Basic Service Set Identifier

INTRODUCTION

Looking back towards past decades the zest for exploring space has only increased with time. First space exploration mission was carried out by the U.S.A as they launched Apollo11 on July 20, 1969, making Neil Armstrong and Edwin Buzz Aldrin the first humans to ever walk the surface of Moon. In the decades to come, we can expect various milestones being achieved in space exploration like humans actually going to other planets. Agencies like NASA, SpaceX plan on sending humans to Mars somewhere in the 2030s. For these missions to be carried out in an efficient way, a system of communication must exist between Earth, satellites, rovers and other robots sent to other planets.

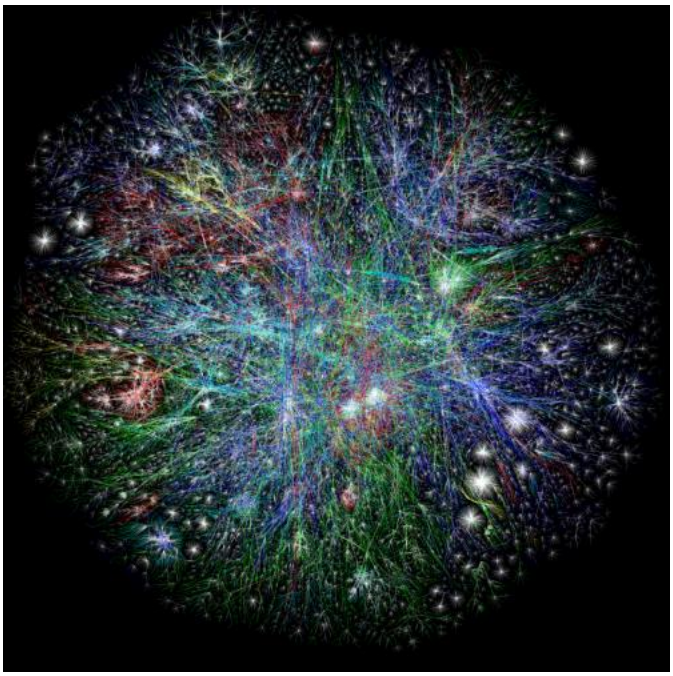

Fig 1. Current status of the Internet on Earth

The figure above shows the present state of the Internet on Earth. Here each color represents an autonomous system. As one can interpret TCP/IP works quite well in terrestrial networks and has advanced a lot over the years. But the communication works differently on Earth and in outer space. The protocol suite followed on earth i.e. TCP/IP is based on a variety of assumptions, assumptions that deep space defies greatly. TCP/IP would work efficiently if it wasn't for long distances, intermittent connectivity, blackouts, high error rates, etc. Considering a scenario in which you want to access Google on Mars.

One will start by typing in the URL address of Google which will be translated to IP address by DNS. A request for translation is sent to the Earth's Deep Space Station (DSS) that is currently in view. The RTT for communication depends on the current positions of planets. The RTT for Earth and Mars last somewhere in 7-40 minutes. Even the greatest delays in the terrestrial network last about 250ms. Say it took 14 mins for the request to be handled and in the meantime as planets are constantly in motion and to add to that they rotate as well so, the DSS could have changed resulting in a different IP address [21]. All the efforts put in until this point are now futile. To operate in this disrupted and delayed environment a new mechanism of Delay Tolerant Networking (DTN) was developed.

1.1 Introduction to Delay Tolerant Networks (DTN)

DTN is a methodology developed to operate in challenged environments where there exists intermittent connectivity, large propagation delays, blackouts, multiple retransmissions, etc. Unlike TCP/IP, it is independent of end to end connectivity and works as an overlay network on the existing TCP/IP protocol stack with the store and forward routing mechanism where the intermediate DTN nodes act as stores for the bundles that are being sent to the next node. Bundles are kept until the successive node doesn't send the custody of the data transmitted. DTN is not just limited to deep space but is also utilized on Earth for military networks, underwater environments or any other challenged environment where TCP/IP struggles.

Various protocols have been developed to specifically operate in delay prone environments like Bundle Protocol (BP), Licklider Transmission Protocol (LTP), Consultative Committee for Space Data System (CCSDS) File Delivery Protocol (CFDP), Saratoga, etc. The protocol that is widely used among these is the Bundle Protocol. Working and conceptualization of Bundle Protocol can be found in [14]. It sits at the Application Layer of internet model and interacts with the native protocols via Convergence Layer Adapters (CLA). The data is transmitted as bundles over intermittent connections by using store and forward routing mechanism. It is the protocol that differentiates conventional TCP/IP stack and DTN stack. LTP provides a way to achieve reliability in communication so it is favorable to use it when UDP is being utilized at a lower layer.

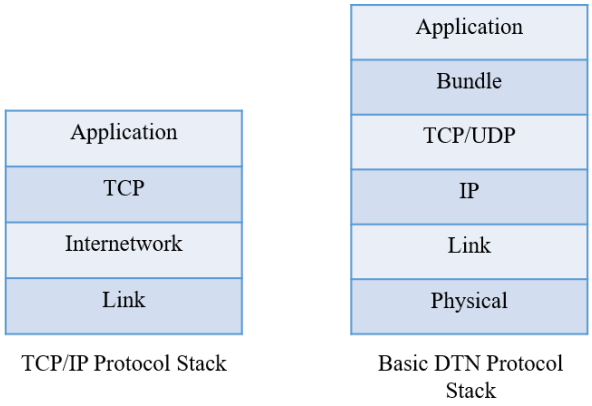

Fig 2. Comparison between TCP/IP and DTN protocol stack.

1.2 Basic Idea and Outline of the Thesis

Communication in Space is a lot more challenging than on Earth. DTN works well in a challenged and stressed environment which has led to the creation of a new segment in communication technologies. With more advances in technology, more prone it becomes to cyber attacks. As the famous saying goes “Change is challenging. And security is like a moving target, so make sure you are able to deal with it and work through frequent changes”. Here our change is DTN that needs to be protected against cyber attacks.

Cyber attacks can cause a lot of damage no matter where they happen. Imagine a space shuttle being hacked, not only is the billion-dollar space shuttle compromised but the scientific data for which the space shuttle was sent in the first place is now facing a threat as well. If threats on Earth are dreadful then the threats in Deep space are catastrophic. The damage inflicted by them is gigantic.

So, this thesis focuses on such security threats that pose a risk of jeopardizing the Interplanetary missions. The aim of this thesis is to verify whether the attacks possible on Earth are also possible in deep space as even if the environments aren’t alike but they still use the same underlying protocols that might end up making them vulnerable just like they do so on Earth. If the same attacks can be done there as well it would be a very troublesome case for the scientists as there is so much you can do by being a million miles away. To find a solution to any kind of problem it is important to identify the problem first i.e.

Detection of an attack and only after achieving that a solution can be crafted i.e. Prevention of an attack. The equipment sent to the Deep-Space is very expensive and any kind of threat to it also disturbs the overall performance of the devices connected to it. This is because the devices like satellites, rovers, orbiters, etc. are constantly communicating to each other, guiding them through the mission, for example, if a navigation satellite that navigates the rover on the planet's terrain was attacked, the rover would automatically be compromised as well.

So, in order to validate the doubts presented, the Layer 2 packets were forged and sent to the devices communicating in the network. To determine how adversely communication is affected by the attack two tests were carried out. Test A is a time-based De-authentication attack and Test B is a Contact window exhaustion attack. After carrying out the attack, it was found out that in Test A the packets reach the destination with a delay that is more than the initial theoretical delay and in Test B about 67% of the packets never make it to the destination. This is a huge problem to tackle in space. A detection mechanism is also proposed that detects if a De-authentication attack is happening on the node and displays the BSSID (Basic Service Set Identifier) of the targeted device and the number of packets sent to it.

LITERATURE REVIEW

In the year 1942 first rocket i.e. V2 missile was launched into space by Germany just to see if it could fly high enough leave Earth's atmosphere to reach space. On October 4, 1957, the Soviet Union launched the first satellite "Sputnik" that placed a radio transmitter in the orbit around Earth. The first space exploration mission wasn't until Apollo11 mission on July 20, 1969, when Neil Armstrong and Edwin Buzz Aldrin first landed on Moon. Now in the 21st century, rockets are launched for finding life forms or possible signs of life on other planets. Along with all other scientific advancements Space Exploration is also becoming an integral part of scientific research. Space exploration has come a long way, now we are talking about Deep Space missions like NASA is planning on sending manned spacecraft to Mars in the year 2020. Earlier in the day Space missions utilized radio signals as means of communication by shooting them towards antennas on spacecraft. Individual missions had their own specialized communication software which led to lack of universality. To overcome this problem a space network that was interconnected and standardized had to be developed. This gave scientists inspiration for the genesis of technology and infrastructure that could support communication in space such as Deep Space Communication (DSN), Interplanetary Internet (IPN), Delay Tolerant Networks (DTN) [17].

2.1 Deep Space communication

Deep Space Communication quite like its name means communicating in deep space with the help of satellites that form a network to aid as a method for communication with each other and to Mission Control Center (MCC) on Earth. The spacecraft sent by NASA is furnished with a communications system that receives commands and other mission-specific information sent from MCC to the spacecraft and returns the scientific data and telemetry acquired during a mission from the spacecraft to Earth. Majority of deep space missions never make it back to Earth. Thus, after launch, the only means through which we can communicate with it is by utilizing its communication systems and tracking equipment [3].

Different types of missions exist in Space Exploration like exploring deep space (planets or ends of the galaxy), climate monitoring using satellites, spy satellites, etc. National Aeronautics and Space Administration (NASA) maintains different missions by operating and maintaining three separate tracking networks. These networks are:

- Deep Space Network (DSN): It is used for missions that require deep space exploration i.e. going beyond Earth. Missions that are operated by NASA or non-NASA agencies are supported by it like the ones that aim at exploring the farthest points in our solar system. The DSN is a network of three ground stations located approximately 120° apart on Earth so that any satellite in deep space is able to communicate with at least one of the stations at all times as there exist time window when communication is interrupted because of the way orbits work i.e. the planet or some other body can come in between satellite and DSS station. The ground stations have to provide software updates, course corrections and a different way to make a scientific observation in case of a query by communicating with satellites [1]. The three ground stations or Deep Space Station (DSS) are located at Goldstone (California), Madrid (Spain) and Canberra (Australia) as shown in Fig 3.

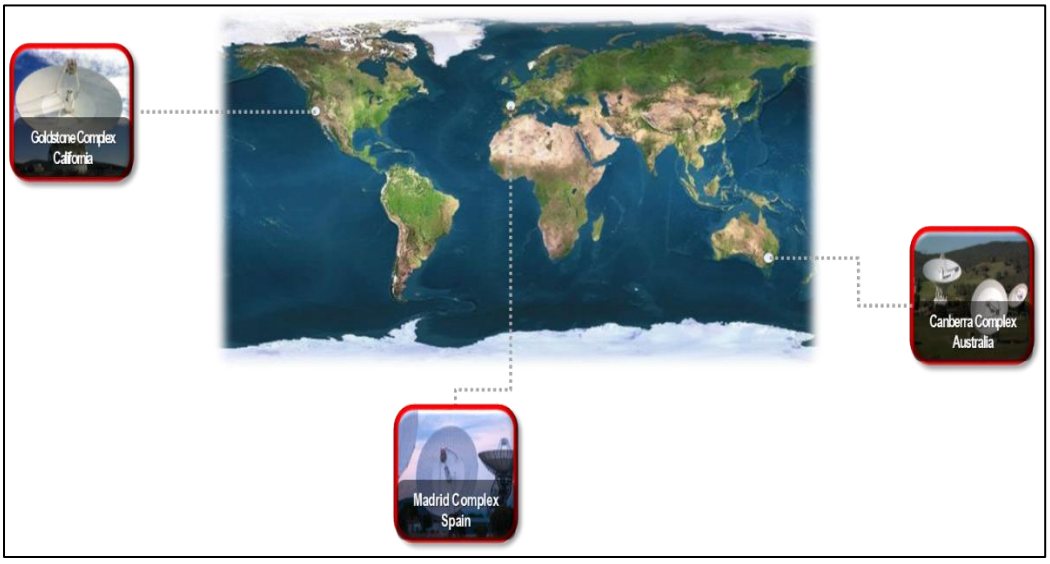

Fig 3. Map of Deep Space Stations

Fig 4 depicts a website developed by JPL so that it can be viewed which DSS station is currently in operation, satellite it is interacting with.

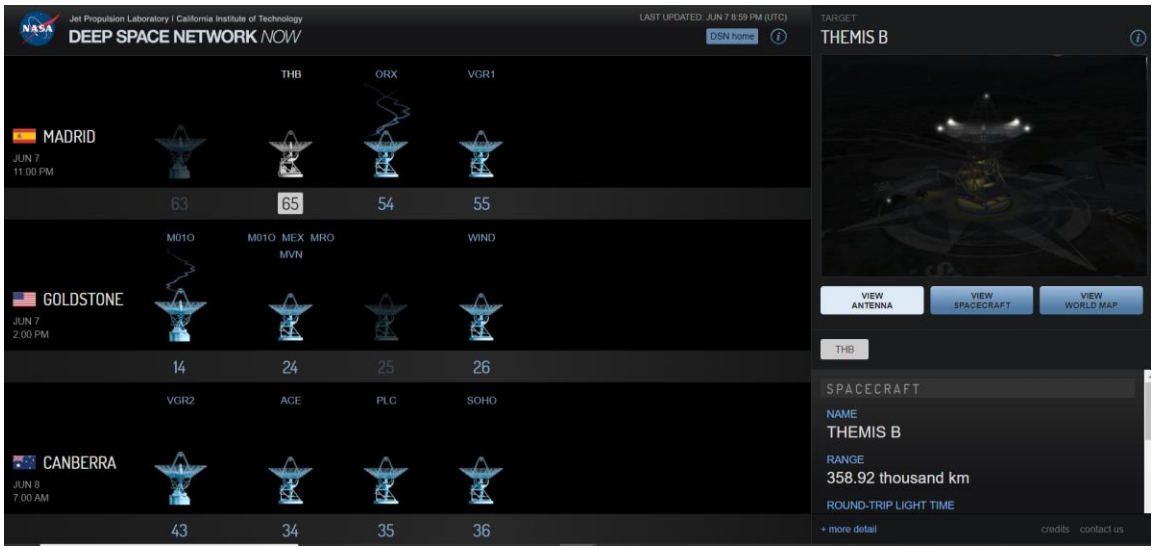

Fig 4. JPL's Deep Space Network Now that shows Currently active DSS [6]

- Near Earth Network (NEN): Non-deep-space missions like tracking, data and communication services in low Earth orbit (LEO), geosynchronous orbit (GEO) highly elliptical orbit, Lunar orbit and missions with multiple frequency bands are supported by it [2].
- Space Network (SN): It is also known as the Tracking and Data Relay Satellite System (TDRSS) which consists of seven geosynchronous satellites and ground stations [17] that act as relay satellites for transmitting data to other satellites. The astronauts in ISS can access wi-fi because TDRSS is acting as a relay, the ground station sends uplink to it which then transmits it to ISS.

2.2 Interplanetary Internet (IPN)

The terminology 'IPN' was first devised in 1997 by Vinton Cerf. IPN is taking internet from Earth and providing it to other planets for efficient transfer of scientific data. It is a network of regional networks [4].

2.2.1 Architecture of IPN

The IPN is a network of regional networks. Interplanetary Backbone Network, Interplanetary External Network, Planetary Network [5] form the key architectural components of the IPN.

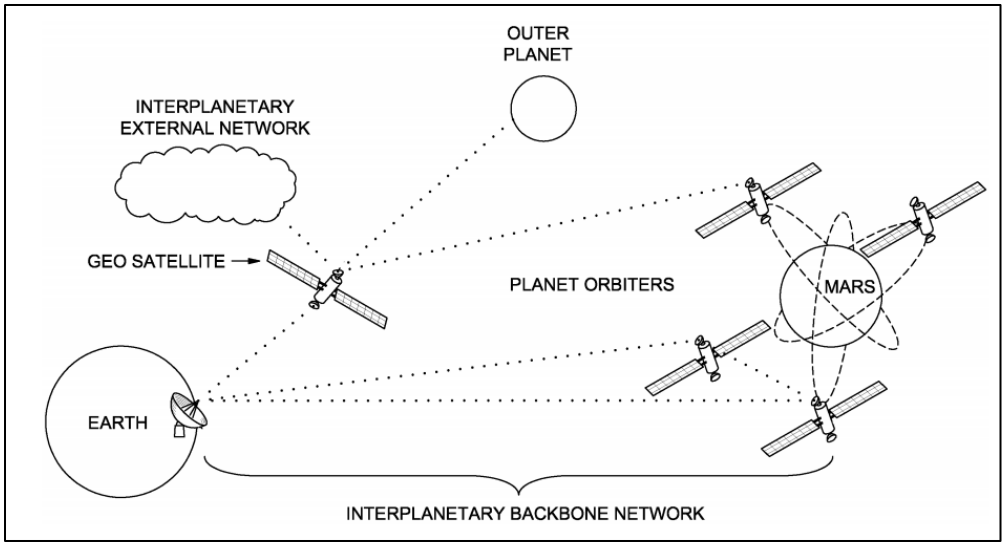

Fig 5. Interplanetary Internet Architecture [30]

- Interplanetary Backbone Network:

  It provides the common infrastructure for communication between earth i.e. DSN, MCC and other planets, satellites, etc. The communication can be direct communication or multi-hop communication.

- Interplanetary External Network:

  Spacecraft flying in groups in deep space between planets, clusters of sensor nodes, and groups of space stations, etc. form this network [5].

- Planetary Network:

  It is the network formed on a specific planetary body by surface and aerial elements. Needless to say, for this scenario to work it requires complete cooperation between aerial and surface elements so that communication can be done efficiently. It comprises of Planetary Satellite Network and Planetary Surface Network.

  - Planetary Satellite Network: It is the network formed by the constellation of satellites around the planet that provides access to the surface elements of the planet to provide commands and navigation.

- Planetary Surface Network: It is the network formed by objects on the surface of the planet like Land Rovers, probes, sensors, etc. It mainly contains two units. The first unit can communicate to satellites, the ones with low transmission capability can communicate amongst themselves. The second unit is Landers and Rovers that acquire data from sensors and transmit it to LEO (Low Earth Orbiting) satellites [18].

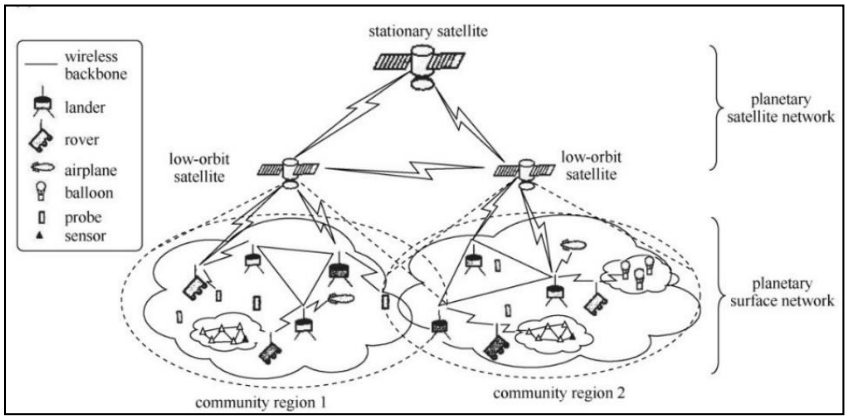

Fig 6. Planetary Network Architecture [5]

2.3 Delay Tolerant Network (DTN)

The term DTN was first coined by Kevin Fall in 2002. It is designed to operate in challenging environments' like a huge distance, intermittent connectivity, blackouts, etc. Kevin Fall came up with a design that merges the idea of IPN to existing terrestrial internet by making some changes to existing TCP/IP protocol stack. DTN acts as an overlay network on existing terrestrial internet comprising of DTN nodes (nodes that participate in DTN). DTN nodes work on the policy of store and forward i.e. in case of link loss the data won't get forwarded but will get stored in the nodes until link revives. DTN defines an abstraction layer below the application layer and on top of the transport layer, called Bundle Layer [7].

Several protocols are added in the blue book of CCSDS to support deep space communication by embedding them in the terrestrial protocol stack. Some of the protocols developed are Bundle protocol, LTP (Licklider Transport Protocol), CFDP (CCSDS File Delivery Protocol), etc.

Bundle protocol is one of the most important protocols that distinguishes the conventional TCP/IP stack from DTN based protocols. The data units transmitted across the network are called “Bundles”. It provides end to end communication by introducing multiple hops in the pathway.

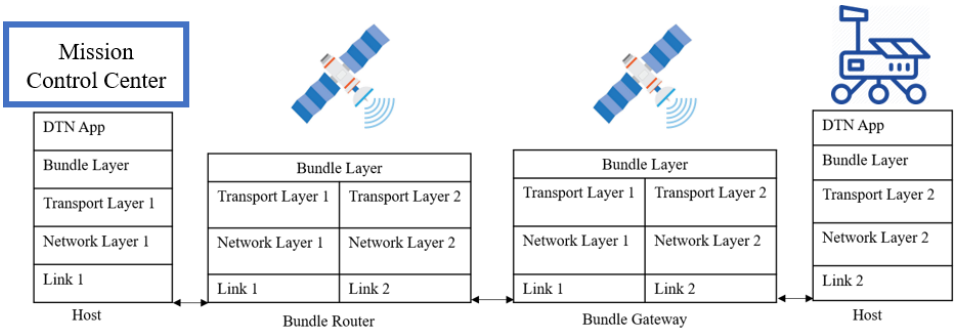

Fig 7. DTN Protocol Stack in communication

2.4 Difference between DTN and Terrestrial Internet

The terrestrial TCP/IP suite cannot work in deep space as they are built of a variety of assumptions like:

- End-to-End connectivity: Between two communicating nodes the link will always be present and the data packets will not get lost in the transmission.
- Round Trip Time (RTT): The RTT won’t be large like the order of minutes or hours, Depending on where the device is the RTT varies. In satellite-based communications RTT of a few milliseconds exists for example communication with a satellite in Geosynchronous Orbit causes a RTT of around 500ms.
- End to End Path: Path between endpoints always exists and the Routing mechanisms will find the single best route. If a node cannot find the path ahead the packets will simply be dropped.
- Error Rates: The error rates and bandwidth asymmetry are next to none.
- Minimal Delays: The distance between the two farthest endpoints on Earth is nothing compared to interplanetary or intergalactic distances, so, the delay caused by earthly distances is very minimalistic like the order of milliseconds. The satellites having an altitude of about 35,000 km operating in Geosynchronous Orbit have a delay of about 240-280 ms.

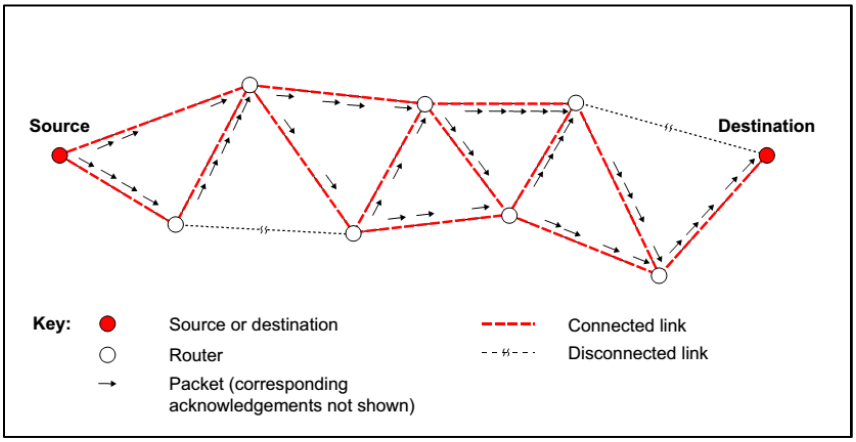

Fig 8. Packet traversal from Source to Destination in Terrestrial Internet [27]

On the other hand, the Deep Space missions defy each of these assumptions. They usually operate in an environment where there are:

- Very Long Propagation Delays: Links in deep space communication may have awfully long propagation delays as the sheer distance ranges to thousands to millions of miles. For example, RTT for two-way communication in Earth-Mars network ranges from 7 to 40 mins depending on how badly Earth and Mars are positioned. The delay isn't constant as all the components of deep space network are in motion so the delay changes with there relative position.
- High Link Error Rates: The links in Deep Space communications are intermittent which can lead to more retransmissions and hence can introduce more error in the link.
- Blackouts: Orbital obscuration can cause intermittent connectivity where you lose line of sight because of planetary bodies like its natural satellites, other celestial bodies in motion. This results in link outage for a brief duration where you cannot communicate with some of the devices.
- Bandwidth Asymmetry: The asymmetry in the bandwidth capacities of the forward and reverse channels is typically on the order of 1000:1 in spacecraft missions [5].

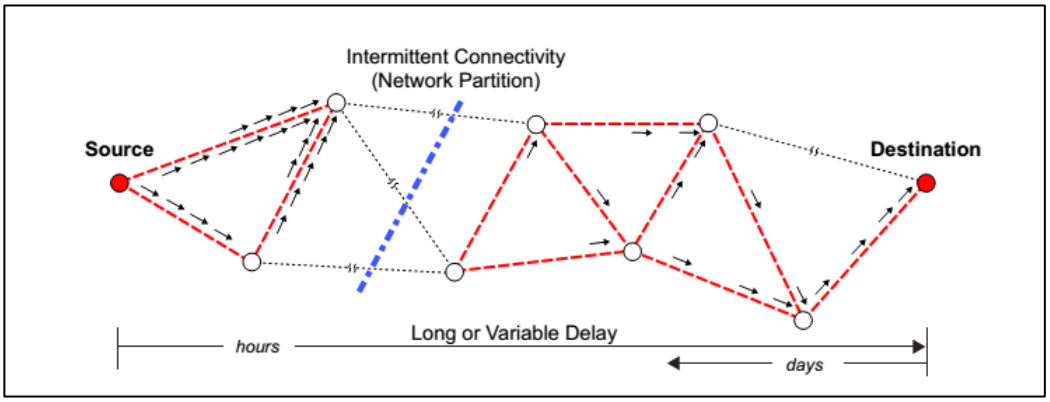

Fig 9. Bundle traversal from Source to Destination in IPN [27].

2.5 Practical Implementations of DTN

2.5.1 United Kingdom-Disaster Monitoring Satellite (UK-DMC)

UK-DMC is a British Earth imaging satellite which is operated by DMCii (Disaster Monitoring Constellation International Imaging). It was manufactured by Surrey Satellite Technology Ltd. (SSTL) and has had three successors.

Table 1. Successors of UK-DMC

Satellite	Launch	Contractor
UK-DMC1	September 2003	N.A.
UK-DMC2	July 2009	Kosmotras
UK-DMC3	July 2015	ISRO

It was the first satellite to demonstrate the use of “Bundle Protocol”. By using the Disruption- and Delay-tolerant networking protocol designed for the IPN it successfully delivered sensor data [9]. As seen from the table there has been a total of three satellites that have been launched into space through the course of approximately twelve years. The first image of South Africas Good Hope was sent in fragments to the Bundle Agent in SSTL’s Ground station. The fragments were then sent to NASA's Glenn's agent using the Internet to resemble them into an image.

2.5.2 DTN implementation using Epoxi Spacecraft

NASA's Jet Propulsion Laboratory in Pasadena, Calif., used Disruption-and Delay Tolerant Networking, to transmit dozens of space images to and from NASA's Epoxi spacecraft back in 2008. Epoxi spacecraft was then located about more than 32 million kilometers (20 million miles) from Earth [10]. Epoxi spacecraft was originally sent to encounter Comet Hartley but was also utilized in a Deep Impact network Experiment (DINET). It was used as Mars data-relay orbiter. The network constituted of ten nodes, nine nodes being on Ground (at JPL) and other being Epoxi. The experiment lasted for about a month in which over 300 images were sent from different JPL nodes at the ground. Experiment's timeline is summarized in the table below [17]:

Table 2. The course of the DINET Experiment

Date	Accomplishment
October 18, 2008	• ION was successfully uploaded on Epoxi spacecraft. • Data was sent to and received from the DINET Experiment Operations Center.
October 20, 2008	• 23 of the images sent to Epoxi spacecraft were transmitted and successfully received at JPL in a course of 3 hours.
October 22, 2008	• During the second pass of the experiment, about 264KB were successfully delivered resulting in about approximately 97.6% link utilization.
November 3, 2008	• 1587.42 KB or 35 image files were delivered via the IPN on 5th DSN tracking pass to the DINET Experiment Operations Center via image reception software.

2.5.3 DakNet

DakNet is a software made to provide connectivity in secluded areas like rural areas. It is a store and forward wireless ad-hoc network. The internet nowadays can provide a speed of up to 22 Mb/s. By using antennas and repeaters range of wifi can be increased and they can be made to work in rural areas.

DakNet combines present transportation infrastructure with wireless data transfer to provide connectivity by using hubs like Post Offices, Cyber Cafes. It transmits digital data to mobile access points (MAPs) i.e. kiosks and portable storage devices mounted on and powered by a passenger bus by using short point to point links that are [17].

2.6 De-authentication Attack

Wireless networks are prone to a variety of attacks and de-authentication is one of them. In wireless communication, the devices must know who they communicate with before communication starts. 802.11 contains mainly three types of frames i.e. Control, Data, and Management.

- Control Frame: It assists in the delivery of Data and Management frames.
- Data Frame: These are the frames that hold the data.
- Management Frame: These are the frames that manage the communication of devices with access points. It comprises of a variety of frames like beacon, Association request and response, Probe request and response, Authentication request and response, De-authentication and Reassociation frames.

One of the intriguing frames in Management frame is “De-authentication” that is used by the device when it wants to break off the connection with the access point. This is a very convenient frame for attack if used correctly. If one manages to spoof addresses and send forged frames to devices connected to the network, it can trick the device into disconnecting from the network. Even if the session was established with Wired Equivalent Privacy (WEP) for the privacy of data, the attacker only requires the MAC address of victim (which can be acquired using wireless network sniffing) as the protocol has no requirement of encryption in this frame [20].

The De-authentication attack can be used in a variety of ways like disconnecting UAV from a device, breaking the connection of two legitimate devices. After successfully implementing the de-authentication attack the attacker can launch more malicious attacks, connect itself to your device, plant virus in it, etc.

2.7 Interplanetary Overlay Network (ION)

Interplanetary Overlay Network or ION as stated in [25] is a software program developed by JPL to test out Delay Tolerant Networks as described in [12]. It implements BP as per [14], LTP as per [13] [31] [17], CFDP as per [28], etc. from DTN protocol stack.

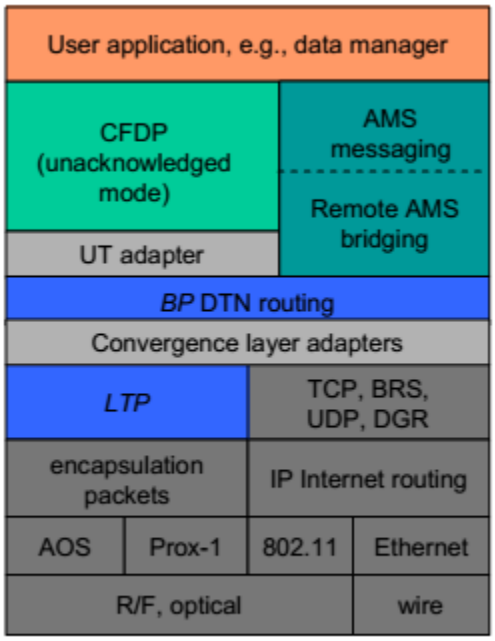

Fig 10. DTN protocol Stack

There are a variety of constraints on the implementation of these protocols because communication between spacecraft and MCC are all wireless and the speed of transmission is very slow because of huge distances.

After installing ION into the systems to utilize it one needs to create configuration files according to requirements of the network. Configuration files contain all the information about the network like contact windows, maximum data to be transmitted, protocols used for transmission, etc. It uses Contact Graph Routing to route the path of the bundles.

It provides various provisions like sending bundles to a node, receiving bundles from a node, Chatting between nodes, keeping time, various kinds of adapters for various protocols, etc. It has its own endpoint naming scheme i.e. '*scheme_name:scheme_specific_part*'. Scheme specific part comprises of two more

subdivisions: '*element_number.service_number*'. An example of such a scheme is *ipn:1.0.* There can be multiple endpoints for a single node each doing a variety of tasks. All the operations run on ION get documented in '*ion.log*' file. With the help of this log file, one can easily keep track of operations done using it and identify errors in them.

RESEARCH GAPS

DTN is bringing us a step closer in carrying out manned missions. The biggest challenge in carrying out Deep Space missions is how to communicate with somebody that is thousands of light-years away from the communicating device. So, to solve this problem a set of protocols were developed that help in the accomplishment of this task. Researchers have managed to successfully implement protocols that aid in DSC, but the question of security still remains to be unanswered. Even though the protocols themselves are integrated with security features [15] [16] they still remain quite vulnerable to attacks. There are currently 4,900 satellites orbiting Earth and some of the satellites are aiding the others like TDRS is used to provide wifi on International Space Station (ISS). In an event of an attack on any of them, the performance of others is hindered as well. The communication in space is co-dependent so to protect the one you have to protect all.

Just like every other technology DTN has its vulnerabilities as well. Deep space follows the store and forward architecture where the satellites in between act as storage nodes and attack on them will lead to a huge amount of data loss. The satellites and rovers on planets communicate via a wireless network so they are prone to a variety of attacks like DoS [24], Spoofing [23], Man in Middle, Blackhole and Greyhole attacks [22]. For wireless communications, before the communication sessions instigate the devices must know with whom they are communicating. As scientific equipment is highly expensive and carries a huge amount of data they need to be protected well against attacks. The purpose of this thesis is to verify whether the attacks possible on Earth are also possible in deep space as even if the environments aren't alike, but they still use the same underlying protocols that might end up making them vulnerable. If the same attacks can be done there as well it would be a very troublesome case for the scientists as there is so much you can do by being a million miles away.

3.1 Objectives

Following points state the objectives of this research work:

- To design a testbed that can support Deep-Space communication.

- To successfully implement communication between nodes in a deep space environment using the ION (Interplanetary Overlay Network) software module.
- To launch De-authentication attack and analyze its effect on Deep Space Network existence.

DESIGN AND IMPLEMENTATION OF DEEP SPACE NETWORK

Deep space is a challenging environment where accomplishing communication in itself is a very problematic task but adding security to it just takes it a step further. In this thesis, the aim is to explore the security parameters in DSC. Until now it has been seen that attacks like DoS, Spoofing, Man in Middle, Greyhole and Blackhole attacks have been done to test the security of the network.

The approach derived for this research work is testing the security of the network by forging layer 2 packets and using them to trick the node into disconnecting from the genuine network. In a wireless network, to establish an initial association Management frames are utilized. Among the major attacks, one is forging the de-authentication management frames and sending them to devices communicating with each other in order to disconnect them. After accomplishing this the attacker can launch any further attacks to take control of the device or damage it permanently. To perform this, a python code was developed that detects the SSID, BSSID and Channel number of the network and launches the attack by forging Layer 2 packets and sends them to the connected users. Entire network or even one device can be targeted if you know the address associated with it.

Fig 11 depicts how an attacker launches the attack on the network. Once the devices are disconnected the attacking machine can steal or corrupt the scientific data acquired or can even do more horrendous things. In this thesis, to determine how adversely communication is affected by the attack two tests were carried out. Test A is a Time-based De-authentication attack and Test B is a Contact window exhaustion attack. Tests are carried out on a two-node testbed where nodes are communicating by using DTN protocols. The communication is monitored under normal circumstances first. After getting the results they are analyzed. Initially, the nodes transmit data and in the midst of that, an attacking machine is used to launch an attack. As the attack is being launched the two test scenarios are executed and their results are documented and analyzed. A detection mechanism is developed using python that detects the de-authentication attack and gives the MAC (Media Access Control) address of the targeted device and number packets sent.

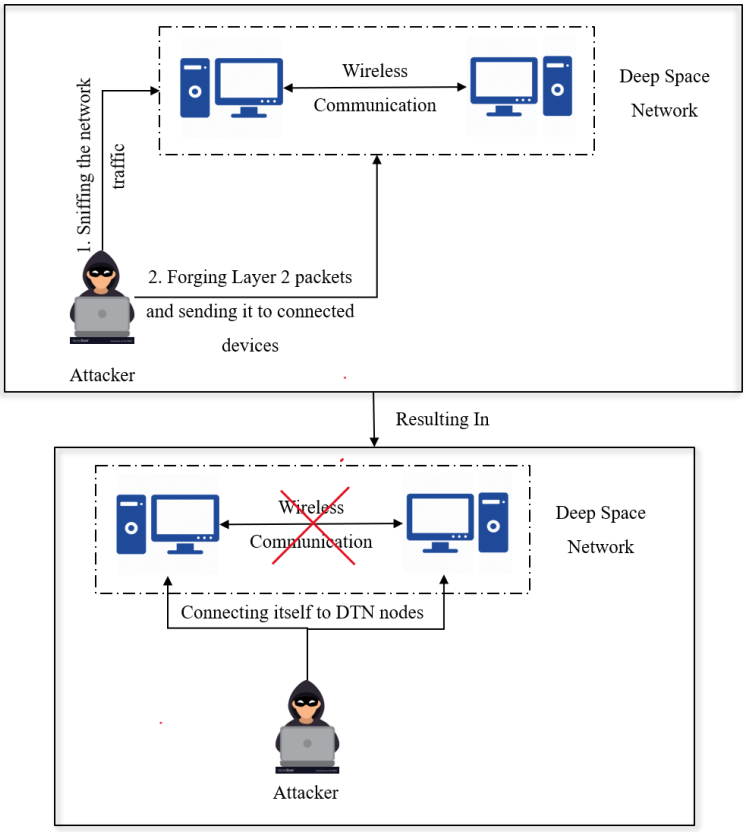

Fig 11. Graphical representation of Attack on Network

In Test A the network is attacked only for the 30 s to see how much delay is caused in communication due to this. As the DTN architecture follows the store and forward architecture attacked nodes don't actually drop the bundles but store them. So, bundles do reach the destination but with a delay. In Test B, the node is attacked until the time of contact in the contact plan is exhausted. In deep space, nodes are in contact for a certain period of time as planets and satellites are in constant motion so there are bound to be periods when they are not in sight of each other. In the configuration designed Node 1 and Node 2 remain in contact for 3600 s or 1 hr with each other. For the sake of reducing the attack time, the contact window time is reduced to 1800 s or 30 min. So, now the attack on the network lasts for roughly 30 mins. The attack starts after transmitting some bundles to ensure bundles were being transmitted before the attack. For this research work, five codes are used. Pseudocode 1 and 3 find SSID, BSSID, and Channel number, Pseudocode 2 finds clients of a specific access point, Pseudocode 4 launches the deauthentication attack and Pseudocode 5 detects the de-authentication attack.

Pseudocode 1: Finding wireless SSID

Output: List with SSID, BSSID and Channel No. of the Access Points in range.

Import all the necessary modules and libraries i.e. Socket, Struct, Shelve, Sys, Traceback.

If some previous data exists in wireless_data.dat file then:

Create a shelve type variable s and open the file.

Use keys() function to extract the information and store it in keys.

Create an empty list.

Add value from keys into list.

Sort the list.

Implement above two steps until the keys isn't empty.

Print the keys.

Close the shelve object using close() function.

If the wireless_data.dat file is empty then:

Create a raw socket named sniff that uses AF_PACKET and accept all ethernet frames by using a socket.socket() method on the server side.

Bind sniff to your monitor mode interface using the bind method.

Create an empty list named ap_list.

Open wireless_data.dat file.

Use sniff to receive information of x bytes in variable fm1.

Extract radio tap length from fm1 in a variable radio_tap_length.

If beacon frame exists in fm1 then:

Calculate source address using radio_tap_length and store it in variable.

If source address is not in list of ap_list then:

ap_list ← source address.

Use radio_tap_length to calculate SSID and Channel and store them.

Change source address to BSSID format and store it.

Create an empty list i.e. list_val.

Print BSSID, Channel and SSID.

list_val ← bssid.

list_val ← channel.

list_val ← ssid.

Transfer the values from list_val to ap_list sequentially.

Close shelve and sys.

Pseudocode 1 provides the SSID, Channel Number and BSSID of the access points in range by sniffing the network for beacon frames. After capturing them the information about wireless connection is extracted from them and placed in a .dat file i.e. “wireless_data.dat”. Data from the previous execution can also be viewed. This code uses a raw socket to work through the frames. It is highly essential to find this data as they will provide the information that is required for forging the packet and then sending that forged packet to the Access Point (AP).

Pseudocode 2: Finding devices connected to specific Access Point.

Output: MAC address and name of Access Point.

Import all the necessary libraries like scapy.

Create a variable and store the name of the interface on which monitor mode is enabled.

Create an empty list to store names of clients of that access point.

Ask the user to enter name of the access point to be monitored in ap_name.

Create a function probesniff():

If the frame has 802.11 probe request frame then:

Extract name of the client from the frame and store it in client_name.

If client_name == ap_name then:

If address of client is not in list then:

Print client_name.

Print MAC address of client.

Add address to list.

Use sniff function with monitor interface and probesniff function.

Pseudocode 2 provides MAC addresses of the devices attached to a particular access point. You need this information in case you aim at targeting a particular device in the network. It does not use the raw socket but uses Dot11 frames to provide the required data. The user

Pseudocode 3: Finding hidden SSID of an Access Point.

Output: SSID of the Access Point.

```
Import all the necessary libraries like scapy.
Create a variable and store the name of interface on which monitor mode is enabled.
Ask user to enter MAC address of the access point to be found and store it in a variable
bssid.
Create a function probesniff():
        If frame has 802.11 probe request frame then:
            If MAC address entered by user is same as that in frame then:
                Print client_name using frame.
Use sniff function with monitor interface and probesniff function.
```

is asked to enter the name of the target. The function probesniff compares the name entered by the user to the one present in the probe request frame. If they match the client name and BSSID is added to the list and displayed.

Pseudocode 3 detects the hidden SSID of the access point. To make the device more secure it is sometimes seen that it is concealed, and the host's device is configured in a way to detect it. In this case, instead of capturing the beacon frame the probe request frame is captured which contains the information about the client. The user is asked to enter the MAC address of the access point whose name needs to be found. Just like pseudocode 2 it uses Dot11 frames and compares the BSSID in the frame with that entered by the user.

Pseudocode 4 launches the attack by sending the forged de-authentication frames. The user is asked to provide a sequence number from the previously created file using Pseudocode 1 that contains the list of access points and provide the targets MAC address. If no address is provided by the user, it is assumed that it wants to attack all the devices in the network. De-authentication frames are created depending on whether the packet is to be sent to the

access point or the client. A function transmits 20 packets at an interval of 10 ms to the specified target.

Pseudocode 4: De-authentication attack.

Input: Sequence ID of target, BSSID and SSID of target.

Output: Output of wireless_data.dat file and packets sent to target.

Import all the necessary modules and libraries i.e. Scapy, Shelve, Sys, Os, Thread.

Create a function for_ap(frame, interface):

    while true do:

        Send 20 forged deauth frame created for access point at an interval of 10 ms.

Create a function for_client(frame, interface):

    while true do:

        Send 20 forged deauth frame created for client at an interval of 10 ms.

Create a function main():

    Create a variable and store the name of interface on which monitor mode is enabled.

    Create a shelve type variable s and open the file.

    Use keys() function to extract the information and store it in keys.

    Create an empty list.

    Add value from keys into list.

    Sort the list.

    Implement above two steps until the keys isn't empty.

    Print the keys.

    Close the shelve object using close() function.

    Ask the user to enter sequence number of targeted access point and store it in a variable.

    Ask user to specify MAC address of target device.

    If victim provides no MAC address then:

        Use broadcast address i.e. FF:FF:FF:FF:FF:FF.

    Give same channel number to yourself as that of access point using system function.

Store the BSSID of target into a variable i.e. bssid from wireless_data.dat file.
Create a two deauth frames to be sent to access point and its client using bssid, RadioTap() and Dot11().
If user has provided specific target then:
    Make a thread to call for_ap function.
    Start the thread.
Make a thread to call the for_client function.
Start the thread.

Pseudocode 5 helps in detecting the de-authentication attack. A pre-requisite for this code is that the monitor mode should be enabled on the device. It comprises of two functions, one function uses a raw socket to capture the de-authentication packets and extracts the BSSID from them and then stores the extracted BSSID in a queue and the other function prints out the stored address as a dictionary.

Pseudocode 5: Detection of De-authentication Attack

Output: Series of MAC address of the device being attacked.
Import all the necessary modules and libraries i.e. Socket, Queue, Thread, Counter.
Initialize a Queue i.e. q.
Initialize a Counter i.e. count.
Create a raw socket named sniff that uses AF_PACKET and accept all ethernet frames by using a socket.socket() method on the server side.
Bind sniff to your monitor mode interface using the bind method.
Create a function ids():
    Make q1 global. (q1 ← Global variable)
    Use sniff to receive information of x bytes in variable fm1.
    Extract radio tap length from fm1 in a variable radio_tap_length.
    Extract BSSID from the frames received.
    Store them in q.
Create a function insert_frame():

Get MAC addresses from q1 and store them in a variable mac.

List ← mac.

Update the list and print the dictionary.

Create two threads that call the functions ids() and insert_frame().

The following figure depicts the flowchart of the work done in the thesis.

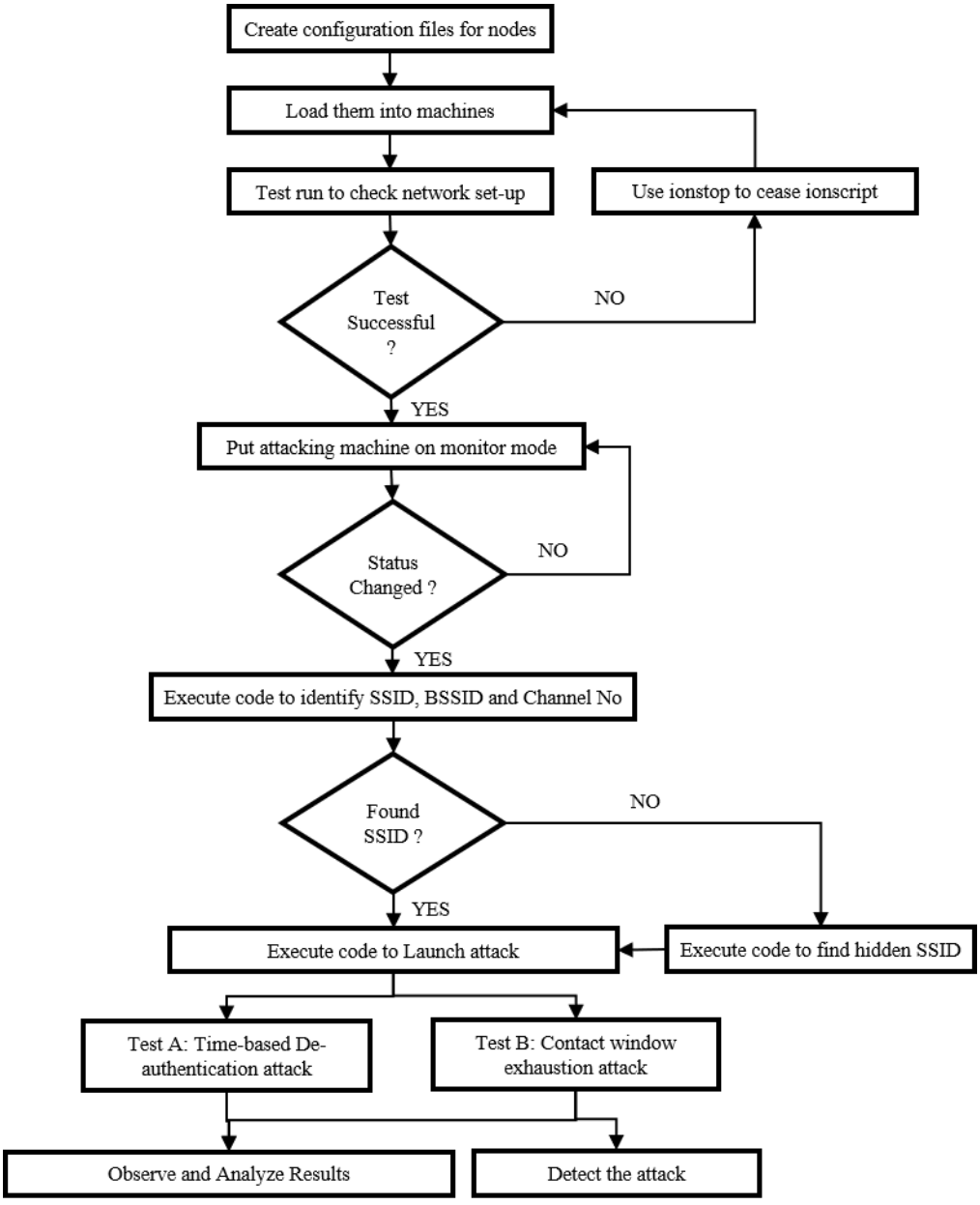

Fig 12. Flowchart of the work done

4.1 Testbed

For the experimental setup, two nodes were configured to support DTN architecture in VMware Workstation. The nodes run on the Linux operating system with ION software installed on them to implement DTN based communication. One of the pre-requisites for using ION is that the Expat XML Parser Library should be installed as well.

Table 3. Configuration details of Nodes

Configuration	Node 1	Node 2
Operating System	Ubuntu 14.04	Ubuntu 16.04
Memory	2 GB	3 GB
Architecture	64 bit	64 bit
Software used for DTN implementation	ION 3.6.2	ION 3.6.2
IP Address	172.16.77.74	172.16.75.159

The testbed consists of two nodes communicating over a DTN network. The network over which they are communicating was designed using a router acting as an access point.

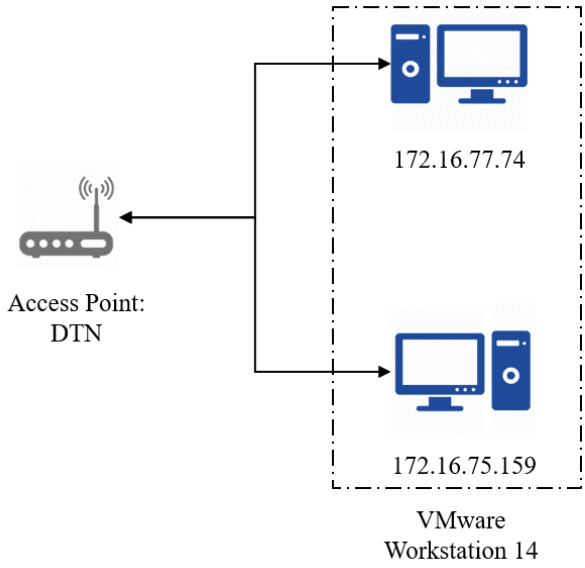

Fig 13. Physical setup of testbed

Fig 13 and 14 show the physical and logical set up of nodes respectively. Here Node 2 is acting as source and Node 1 is acting as the destination. In the physical setup, two nodes are created on VMware. They are connected to a common network "DTN" that is used for transmitting data.

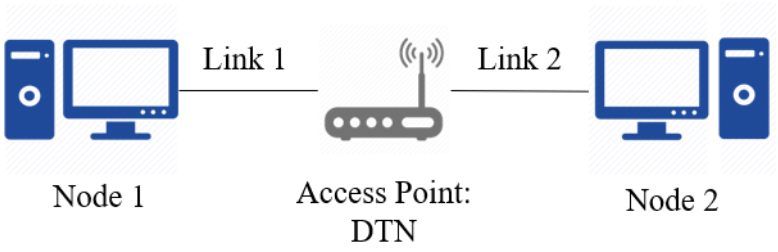

Fig 14. Logical setup of Testbed

To establish communication between two nodes they need to be properly configured so that they know the identities of other nodes, contact plans, protocols used for communication, etc. In DTN exchange of data is very expensive so it is better to pre-place the information at the nodes in the network. The links in deep space are opportunistic but the contacts are scheduled. For example, the satellites have a pre-configured orbital motion that can help to form their contact plans. Therefore, you can use the contact plans to pre-place the information in the configuration files. The contact plans are placed in "*config.rc*" file along with other relevant network information so that each node is set up according to network topology.

ION transmits bundles to nodes via an outduct. If the nodes aren't directly connected ION analyses the contact graph provided and identifies the hops to be traveled to reach the destination. Contact Plans specify the time two nodes remain in contact with each other i.e. the time in which data can be transmitted or received.

In Fig 15. the configuration file of Node 2 is shown. The configuration file is divided into multiple configuration files.

- ionadmin: It specifies contact windows, the data rate for transmission and range i.e. specifies the physical distance between nodes.

- *a contact +1 +3600 74 159 1000000* command states that node with address 74 and node with address 159 stay in contact from 1s to 3600s (60 min) and can transmit data at the rate 10,00,000 bytes.
- *a range +1 +36000 74 74 1* command states the physical distance between the nodes. The nodes stay in contact from 1s till 36000 s. There is a self-link of the node 74 i.e. connection to itself. Data transmission on the link is anticipated to have a delay oft1s to reach the destination also called as One Way Light Time.
- *m production* and *m consumption* command states that the node is to consume and produce a mean of 1000000 bytes/s.

➢ ltpnadmin: It specifies transmission speeds, spans, sessions, block size for LTPCL. Here the LTP is used alongside UDP.

- *1 32 131072* command states that a maximum of 32 sessions can occur and the block size for transmission is 131072 bytes.
- *a span 74 32 32 1400 10000 1 'udplso 172.16.77.74:1113' 300* adds a connection that uses 1400 byte segments and a block size of 10,000 bytes that denotes the amount of data sent in a session. Command 'udplso' is UDP−based LTP link service output task. It is used to establish a UDP based link to itself or to another node. The default UDP port that is being used by LTP has been reserved by IANA as 1113.

➢ bpadmin: bpadmin is ION BP administrative interface. It specifies schemes used for endpoints, endpoint IDs, outduct and induct for node.

- *a scheme ipn 'ipnfw' 'ipnadminep'* command states that add the ipn scheme is used for nodes. Here 'ipnfw' is the forwarding engine and 'ipnadminep' is the administration program.
- *a endpoint ipn:159.0 q* command states add an endpoint on the local node and queue the incoming bundles.
- *a protocol ltp 1400 100* command states that add LTP protocol and estimate the transmission capacity assuming there are 1400 bytes of each frame for payload and 100 bytes for overhead.

- *a induct ltp 159 ltpcli* command states that add an induct to receive bundles using LTP. ltpcli i.e. LTP−based BP convergence layer input task is used to implement the induct command.
- *a outduct ltp 159 ltpclo* command states that add an outduct to send the bundles using LTP. ltpclo i.e. LTP−based BP convergence layer output task is used to implement the outduct command.

➢ ipnadmin: Simply put it contains IPN schemes routing tables.

- *a plan 74 ltp/74* command states that add an egress plan about where bundles are to be transmitted on Node 74 and that they are to be queued on protocol LTP.

```
## begin ionadmin
1 159 ''
s

a contact +1 +36000 74 74 1000000
a contact +1 +3600 74 159 1000000
a contact +1 +3600 159 74 1000000
a contact +1 +36000 159 159 1000000

a range +1 +36000 74 74 1
a range +1 +3600 74 159 1
a range +1 +3600 159 74 1
a range +1 +36000 159 159 1

m production 1000000
m consumption 1000000
## end ionadmin

## begin ltpadmin
1 32 131072
a span 74 32 32 1400 10000 1 'udplso 172.16.77.74:1113' 300
a span 159 32 32 1400 10000 1 'udplso 172.16.75.159:1113' 300
s 'udplsi 172.16.75.159:1113'
## end ltpadmin

## begin bpadmin
1
a scheme ipn 'ipnfw' 'ipnadminep'
a endpoint ipn:159.0 q
a endpoint ipn:159.1 q
a endpoint ipn:159.2 q
a protocol ltp 1400 100
a induct ltp 159 ltpcli
a outduct ltp 74 ltpclo
a outduct ltp 159 ltpclo
s
## end bpadmin

## begin ipnadmin
a plan 74 ltp/74
a plan 159 ltp/159

## end ipnadmin
```

Fig 15. Configuration file for two nodes communicating via LTP/UDP

After creating the configuration files ionstart command is used to load the information about the network so that the communication can commence. A shellcode was developed to send multiple files across the network using bpsendfile command. Similarly, the receiving node contains a shellcode to accept receiving files using bprecvfile. Fig 16 shows a shellcode for sending multiple files across DTN. The code asks the user to enter source

and destination endpoint IDs (EID) and sends 24 bundles at an interval of 10s to the destination node.

```
#!/bin/bash

read -p "Enter the source endpoint: " src
read -p "Enter the destination endpoint: " dest
for i in {1..24..1}
do
        echo "start: $(date + %N)">>startsend.txt
        bpsendfile $src $dest multiplebundledata.txt
        echo "------------------------------------------------"
        echo "File no. $i is sent to destination $dest"
        echo "------------------------------------------------"
        sleep 10
        echo "end: $(date + %N)">>endsend.txt
done
```

Fig 16. Shellcode to send multiple files on DTN network

4.1.1 Preparing the Testbed

- As the nodes are created in a Virtual Environment to access the wireless network they will need wireless adapters. So, connect the wireless adapter to each of the nodes so that the communication can commence.
- To launch the attack, the traffic in the network needs to be sniffed first. To do this the wireless card needs to be in monitor mode where it can sniff the network from an Access Point without being connected to it. To carry out this task another wireless card needs to be attached to the attacking machine as well.

4.2 Testing

There can be ample advancements in technology but if they aren't secure then all of them become minuscule. DTN also uses the same underlying protocols like TCP/IP so, the query was if it was possible to launch an attack that works on the terrestrial network but will also work on DTN. When tested it came to be that De-authentication attack can also work in a DSN.

First, testing of the configuration files created is done i.e. whether they can successfully support deep space communication or not with the help of ION software. The configuration files are loaded into the system so that it becomes aware of the network topology. Now the network formed can be tested by sending a file across the network via bpsendfile or chatting with the other node via bpchat.

4.2.1 Examining the network by sending a file

- Configuration files of two nodes with IP address 192.168.17.132 (Node 1) and 192.168.17.128 (Node 2) are loaded into the system using ionstart command.
- Execute the shellcode to send a file across the network on Node 1 and execute the shellcode for receiving the file on Node 2.
- To check if the file has been transferred "*ion.log*" file can be referred that documents all the operations performed using ION. A file with name "testfile#" will be stored in the directory you are working. Here # represents the number like testfile1.

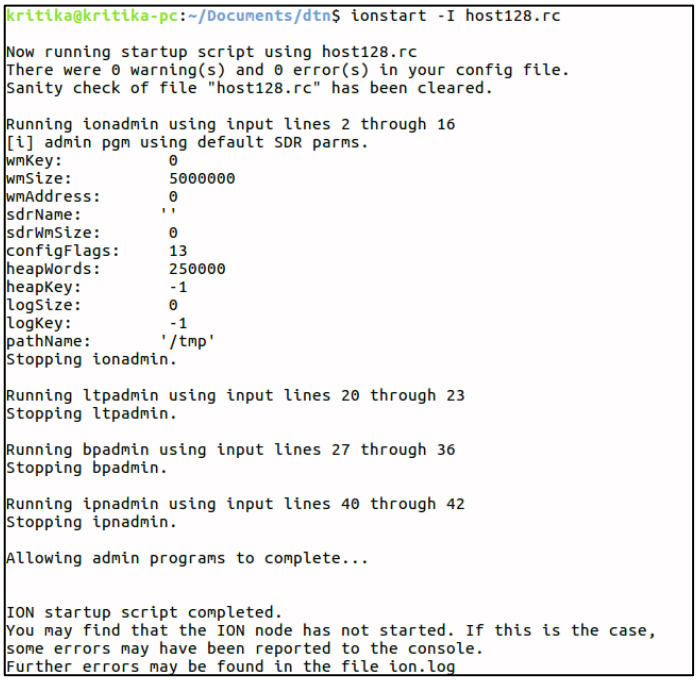

Fig 17. Loading configuration file for Node 2 into the system

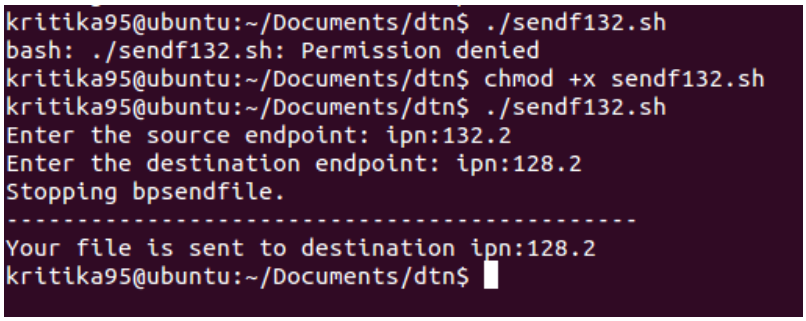

Fig 18. Execution of shellcode to send a file from Node 1

```
[2019/07/01-14:11:50] [i] bprecvfile is running.
[2019/07/01-14:11:50] [i] bprecvfile has created 'testfile1', size 811.
[2019/07/01-14:11:53] [i] bprecvfile interrupted.
[2019/07/01-14:11:53] [i] Stopping bprecvfile.
```

Fig 19. An entry about receiving a file at Node 2 in ion.log file

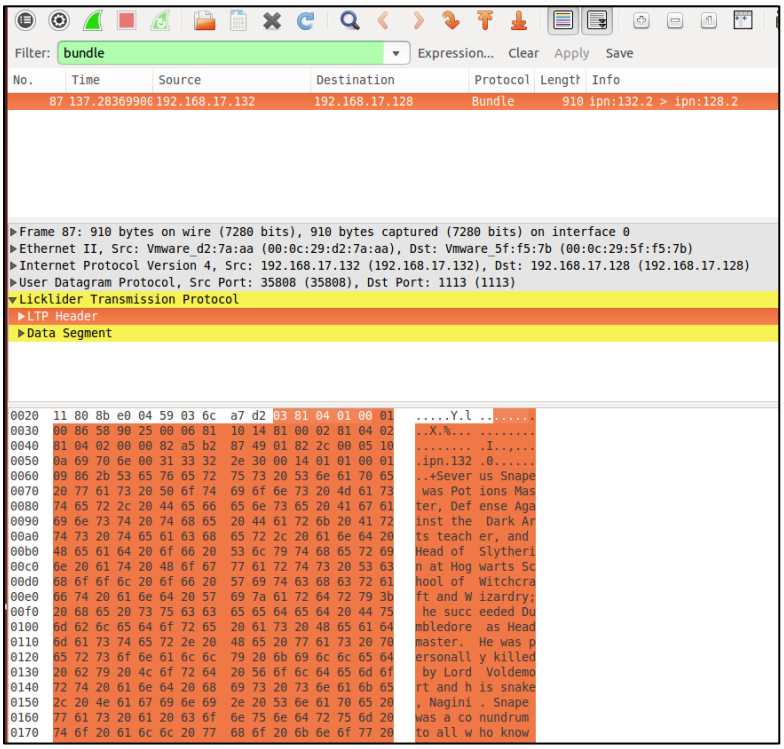

Fig 20. Bundle captured in Wireshark while sending a file from Node 1 to Node 2

4.2.2 Launching De-authentication attack on the network

In this scenario Node 1 and Node 2 are acting as rovers communicating to each other on the Martian surface by using a wireless network. Node 1 is sending data to Node 2 when another spy rover starts launching De-authentication attack on them. The attacking machine in this scenario is working with Kali Linux and a python code is used to launch the attack.

The following series of screenshots show the file transfer between nodes and the traffic captured in Wireshark.

```
kritika95@ubuntu:~/Documents/dtn/Detecting deauth$ ./send_file159.sh
Enter the source endpoint: ipn:74.2
Enter the destination endpoint: ipn:159.2
------------------------------------------------------
File no. 1 is sent to destination ipn:159.2
------------------------------------------------------
------------------------------------------------------
File no. 2 is sent to destination ipn:159.2
------------------------------------------------------
------------------------------------------------------
File no. 3 is sent to destination ipn:159.2
------------------------------------------------------
------------------------------------------------------
File no. 4 is sent to destination ipn:159.2
------------------------------------------------------
------------------------------------------------------
File no. 5 is sent to destination ipn:159.2
------------------------------------------------------
------------------------------------------------------
File no. 6 is sent to destination ipn:159.2
```

Fig 21. Execution of shellcode to send multiple files to Node 2

```
[2019/07/04-15:14:39] [i] : ltpcli() [0x401b51]
[2019/07/04-15:14:39] [i] : /lib/x86_64-linux-gnu/libpthread.so.0(+0x76ba) [0x7fbeb9c9e6ba]
[2019/07/04-15:14:39] [i] : /lib/x86_64-linux-gnu/libc.so.6(clone+0x6d) [0x7fbeb931041d]
[2019/07/04-15:14:39] [i] bprecvfile has created 'testfile7', size 1209.
[2019/07/04-15:14:42] at line 2205 of bp/library/ext/sbsp/sbsp_bcb.c, Assertion failed. (rules)
[2019/07/04-15:14:42] [i] Current stack trace:
[2019/07/04-15:14:42] [i] : /usr/local/lib/libici.so.0(printStackTrace+0x2b) [0x7fbeb95e069b]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libici.so.0(_iEnd+0x21) [0x7fbeb95e0c61]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(sbsp_bcbReview+0x84) [0x7fbeb9a719a4]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(reviewExtensionBlocks+0x4e) [0x7fbeb9a6bd2e]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(+0x217b6) [0x7fbeb9a627b6]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(bpEndAcq+0x122) [0x7fbeb9a639c2]
[2019/07/04-15:14:42] [i] : ltpcli() [0x401b51]
[2019/07/04-15:14:42] [i] : /lib/x86_64-linux-gnu/libpthread.so.0(+0x76ba) [0x7fbeb9c9e6ba]
[2019/07/04-15:14:42] [i] : /lib/x86_64-linux-gnu/libc.so.6(clone+0x6d) [0x7fbeb931041d]
[2019/07/04-15:14:42] at line 1419 of bp/library/ext/sbsp/sbsp_bib.c, Assertion failed. (rules)
[2019/07/04-15:14:42] [i] Current stack trace:
[2019/07/04-15:14:42] [i] : /usr/local/lib/libici.so.0(printStackTrace+0x2b) [0x7fbeb95e069b]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libici.so.0(_iEnd+0x21) [0x7fbeb95e0c61]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(sbsp_bibReview+0x84) [0x7fbeb9a73124]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(reviewExtensionBlocks+0x4e) [0x7fbeb9a6bd2e]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(+0x217b6) [0x7fbeb9a627b6]
[2019/07/04-15:14:42] [i] : /usr/local/lib/libbp.so.0(bpEndAcq+0x122) [0x7fbeb9a639c2]
[2019/07/04-15:14:42] [i] : ltpcli() [0x401b51]
[2019/07/04-15:14:42] [i] : /lib/x86_64-linux-gnu/libpthread.so.0(+0x76ba) [0x7fbeb9c9e6ba]
[2019/07/04-15:14:42] [i] : /lib/x86_64-linux-gnu/libc.so.6(clone+0x6d) [0x7fbeb931041d]
[2019/07/04-15:14:42] [i] bprecvfile has created 'testfile8', size 1209.
[2019/07/04-15:14:49] at line 2205 of bp/library/ext/sbsp/sbsp_bcb.c, Assertion failed. (rules)
[2019/07/04-15:14:49] [i] Current stack trace:
```

Fig 22. Entries in the ion.log file about receiving files from Node 1

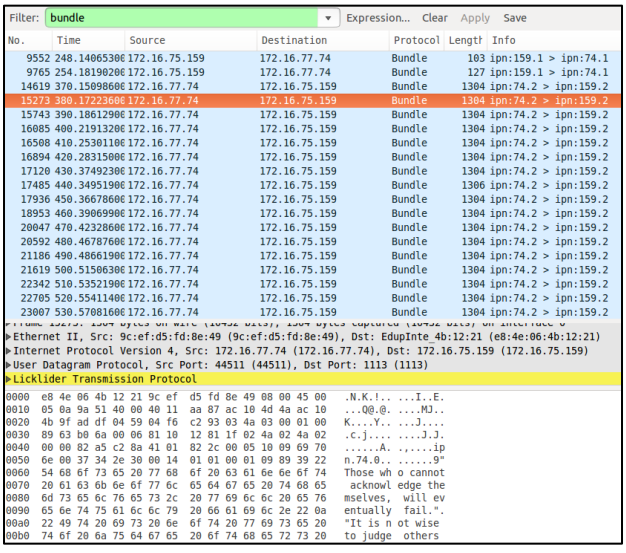

Fig 23. Multiple bundles transmitted across the network as shown by Wireshark

Now that the nodes are running and transmitting data attack will now be launched on them. Attacking machine needs to be equipped with some functionalities before it can launch the attack. The following screenshots depict the series of steps taken to launch the attack and the attack itself.

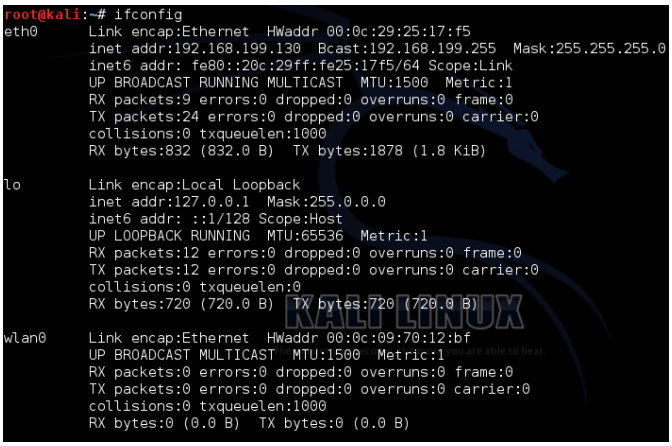

Fig 24. Wireless adapter has been added to attacking machine

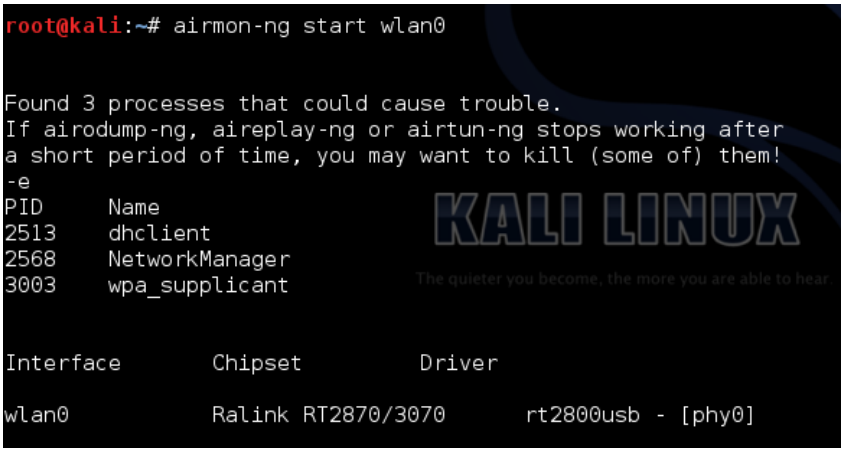

Fig 25. Changing mode of wireless card from Managed to Monitor in order to sniff traffic

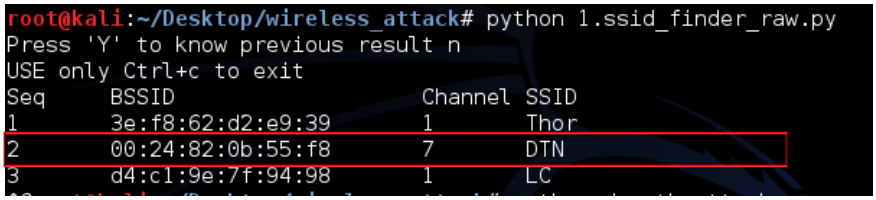

Fig 26. Python code finding out BSSID, Channel and SSID of different Access Points

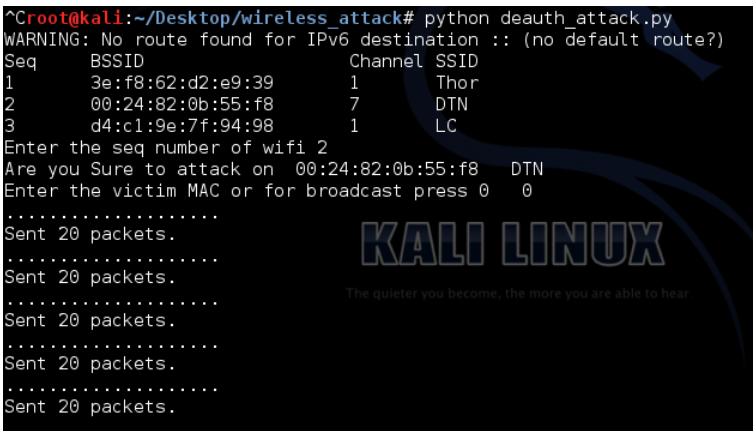

Fig 27. Attacking the nodes connected to DTN using python based De-authentication code

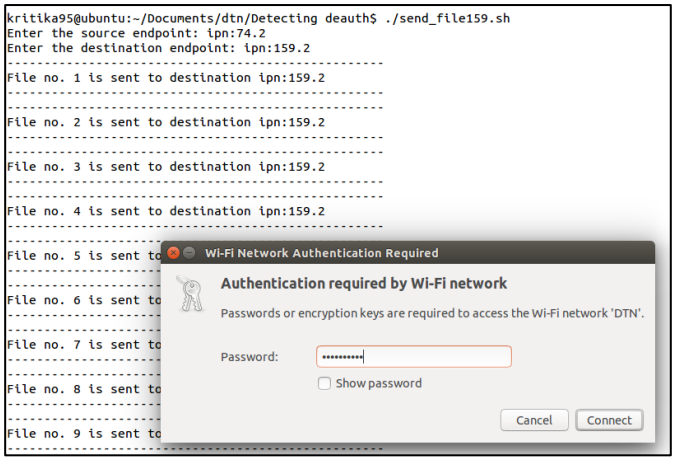

Fig 28. Node disconnects from the network

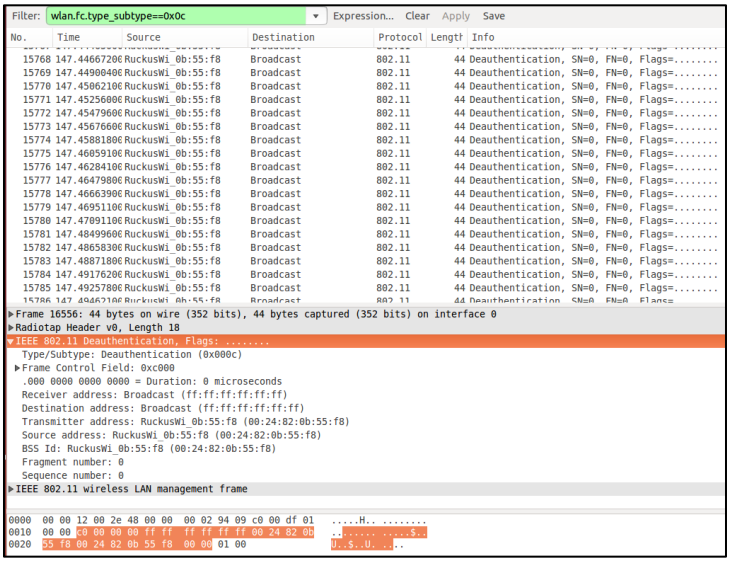

Fig 29. De-authentication packets sent to node captured in Wireshark

4.2.3 Detecting the De-authentication Attack

A mechanism that detects if a de-authentication attack is happening on a node is developed. The IP based attacks aren't easy to identify as they can change after a certain period but attacks on MAC layer are identifiable. A python code that detects the de-authentication attack was executed on one of the nodes connected to the access point. It captures the de-authentication packets and displays the MAC address of the device that is getting attacked and the number of de-authentication packets sent to it.

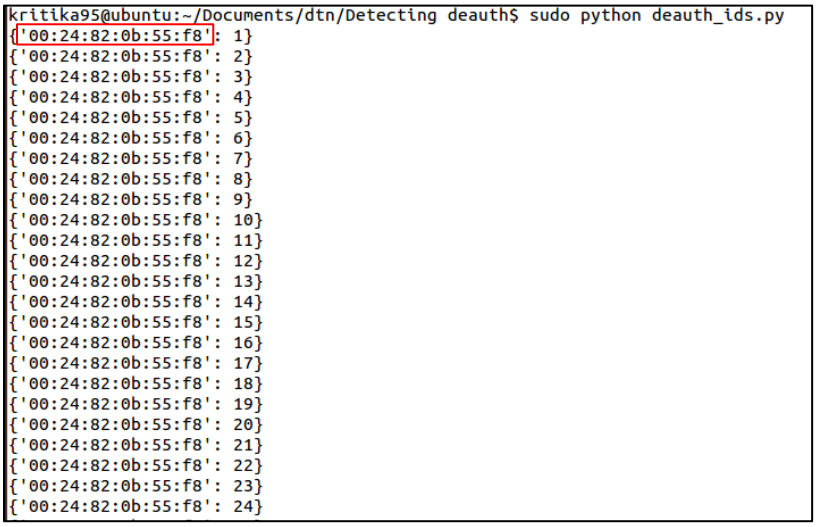

Fig 30. Output of code that detects the De-authentication attack

The highlighted portion in the above figure displays the MAC address of the device that is currently being attacked which in our case is the MAC address of access point.

RESULTS AND DISCUSSIONS

In this section, the results and analysis of the attack done are presented. When the attack occurs, it disconnects nodes from the network leaving them unable to communicate with each other. As the DTN follows a store and forwards architecture so, the bundles are not dropped by the node on the attack but are actually stored in its memory until the link is available again. This might not seem that big of an issue in the terrestrial network but in deep space memory is limited and along with all the other delays if one more delay adds up in the list it affects the productivity adversely.

A variety of tests were performed to observe how badly network gets affected in an event of an attack. In our scenario Node 1 sends a series of 24 bundles to Node 2. In Test A: Time-based De-authentication attack the nodes are communicating as they normally would but an attack is launched in between for just a few seconds to observe how adversely it affects the network. In Test B: Contact window exhaustion attack the contact window of nodes is exhausted for example contact window of 1 hr between two nodes is exhausted by executing the a ttack for 60 mins.

The figure below shows a graphical representation of the time it takes for a bundle of size 55.4kB to be transmitted from Node 1 to Node 2 in an event of no attack.

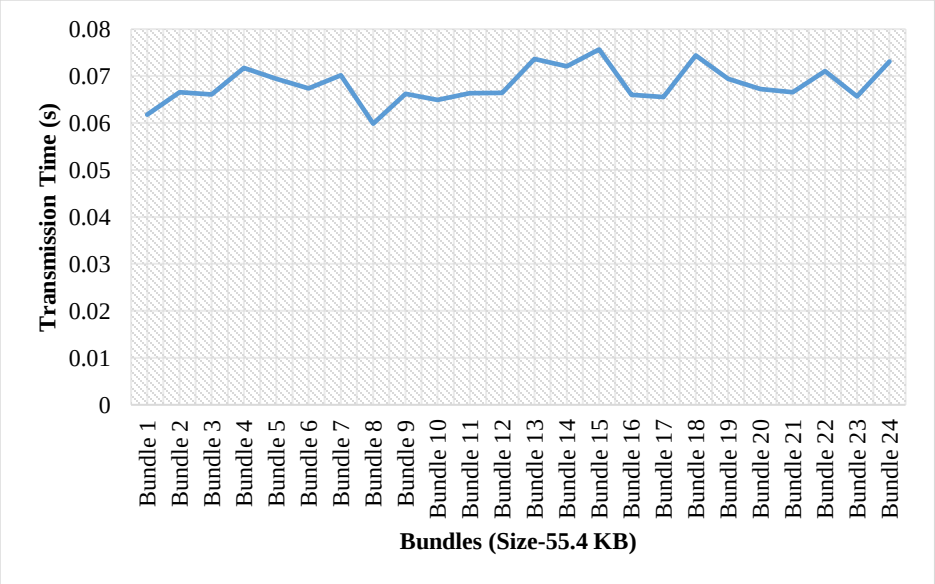

Fig 31. Bundle Transmission from Node 1 to Node 2

For a normal transmission, it takes about 20.25s to complete the transfer from Node 1 to Node 2 so for every bundle transmission time (T_t) is about 0.84s. For the tests carried out, the delays caused by the intermitted links and processing are not considered.

5.1 Test A: Time-based De-authentication attack

In this test, the behavior of the network and the effect on bundle transmission is observed when an attack happens. The attack in this scenario is launched for about 30 s and then stopped so that the node can again connect to the network. After analyzing the bundle transmission values it was found that the delay of each bundle sent after the attack has increased. Initially, the bundles took about 20.25 s to be transmitted across the nodes but after the attack, the delay of 30 s was added to each transfer. The addition of 30 s is a big number in IPN.

The graph below shows the transmission time of the bundles of size 55.4 KB from Node 1 to Node 2. As seen in the graph the transmission until Bundle 7 is normal and as soon as the attack is executed before transmission of Bundle 8 the delay of 30 s is added to each bundles transmission time. If this value is associated with other delays and the total value of delay is calculated there would be a significant increase in transmission time.

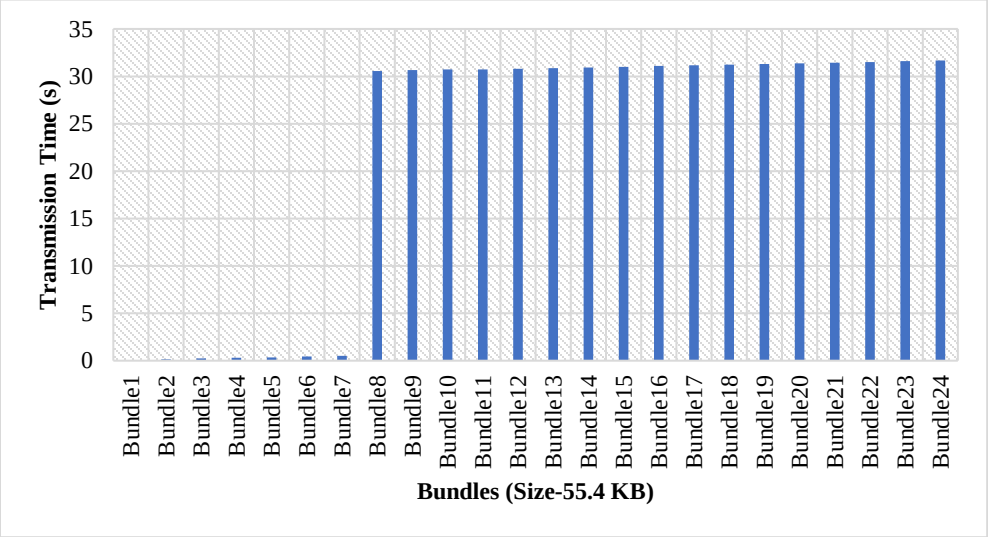

Fig 32. Bundle transmission in Test A: Time-based De-authentication attack

The graph below shows the transmission time for bundles of size 55.4 KB when the normal transmission is taking place and when the attack is taking place. As seen from the graph

the comparison of transmission time for both the scenarios is displayed. In the normal transmission, bundles take less than 1 s and when the attack happens first seven bundles are transmitted normally but the eighth bundle onwards the delay increases immensely for rest of the bundles.

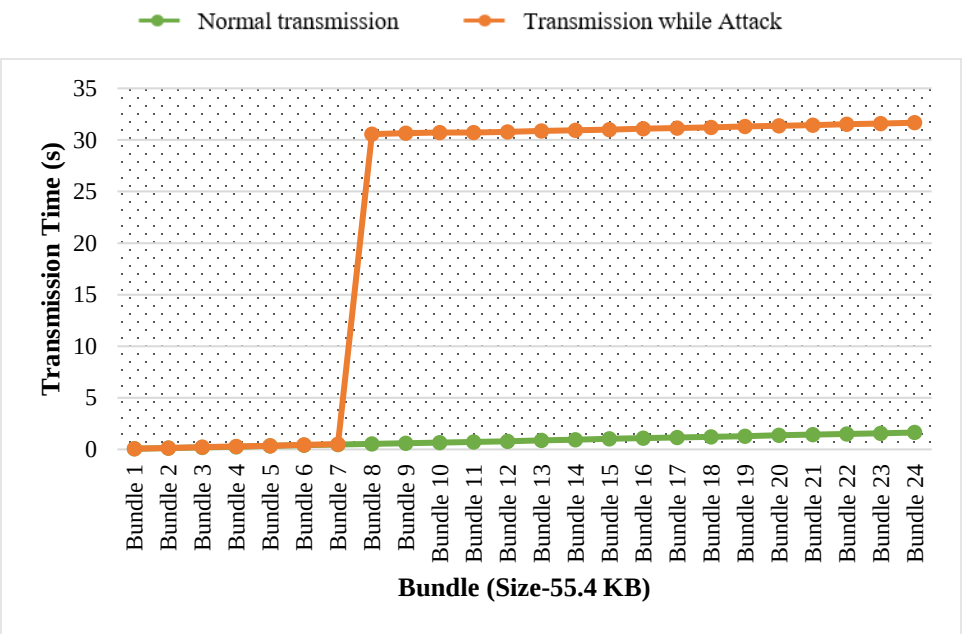

Fig 33. Comparison of transmission of Bundles when normal transmission is happening and when attack happens

5.2 Test B: Contact window exhaustion attack

In this test, the contact time of nodes is exhausted by executing the attack for as long as the contact window lasts. Every pair of a node in Deep Space Communication has a particular contact time in which they can communicate, outside that contact window they cannot communicate. For the sake of ease, the contact window time is reduced to 30 mins from 1 hr. By analyzing the previous experiment, it takes approximately 20 s to transmit all bundles between the nodes. The attack was executed after the ninth Bundle was sent i.e. about 7 s after the transmission started and lasted for about 30 mins.

The graph below shows the transmission time of bundles when the attack is executed. As inferred from the graph the execution until the ninth bundle is as expected but just as the

attack starts the connection between the node and the network is terminated. This results in the inability to transmit the bundles to the destination node.

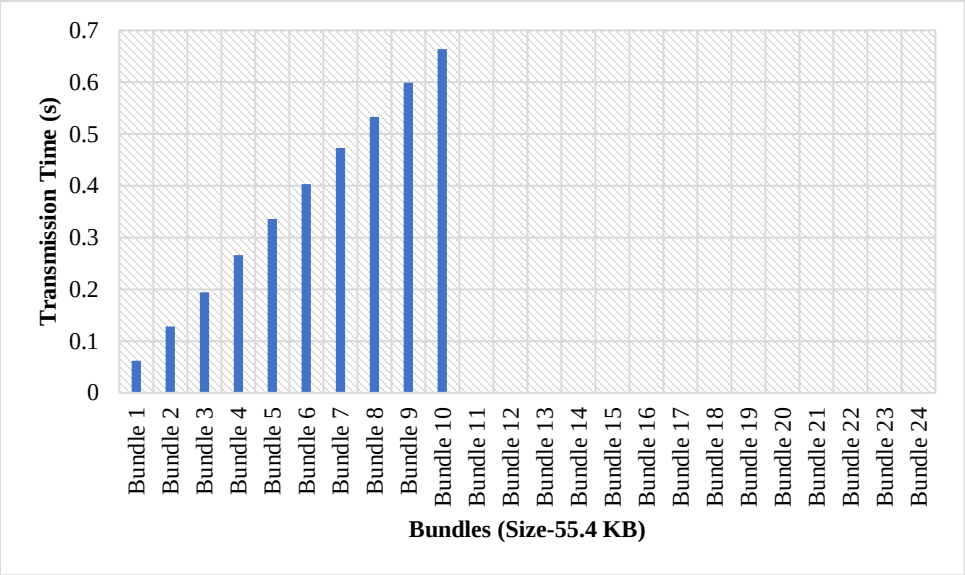

Fig 34. Bundles transmission in Test B: Contact window exhaustion attack

As nodes were not given a chance to reattach themselves to the network the remaining bundles never got delivered to their destination resulting in a huge information loss. In a scenario where this was mission-critical information, it would be catastrophic to not being able to get the data to the destination.

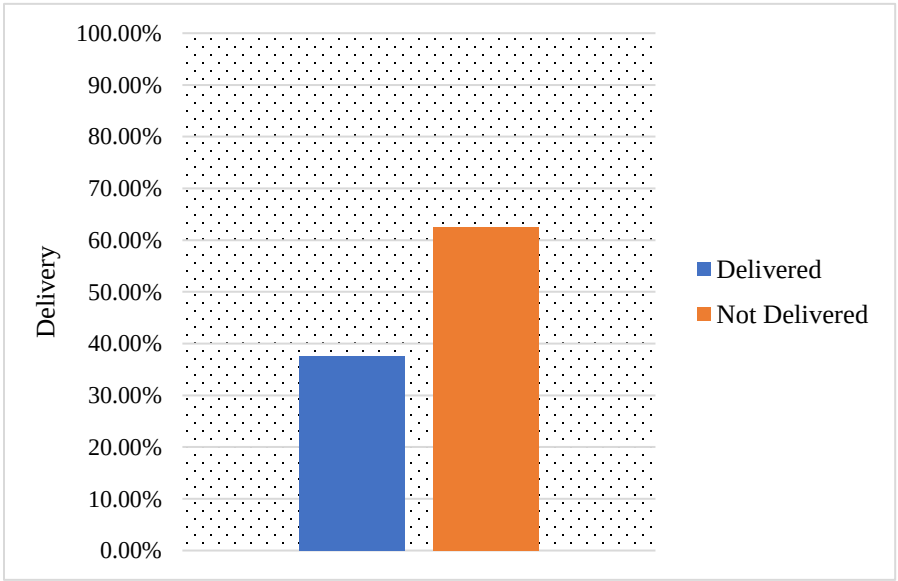

Fig 35. Analysis of bundle delivery due to Attack

The graph above shows the analysis of bundle delivery after the attack happened. If all the bundles were to get delivered that would result in a 100% delivery rate, for example, the scenario in which the bundles were transmitted normally. After analyzing the outcome of the attack on bundle delivery it was observed that the delivered bundles only constituted 37% of the total bundles. As a result, the performance has downgraded immensely because of the attack. In space resources are expensive. Due to limited bandwidth, retransmission of bundles cannot be afforded. In a case where about 67% of the bundles are to be retransmitted is a very costly operation especially in a place where resources are already limited. Consider a network with multiple nodes having a similar scenario as explained in Test B the two nodes that got attacked and their contact window exhausted will have to wait until their next contact window to transmit the bundles. So, the bundles get delayed even more for delivery on top of not getting transmitted in the previous contact window.

5.3 Protection Mechanisms

There is no readily available protection mechanism against de-authentication attack as De-authentication is not a request but a notification. Even if one captures the de-authentication packets in Wireshark it is impossible to determine who the attacker is as the source and destination address in the forged packets is same as that of the machines in the network that are being targeted.

In order to win against an attack of this kind, in 2009 IEEE designed a new variation of 802.11 standard i.e. 802.11w which is Protected Management Frame (PMF) that checks the legitimate client by sending a query frame that confirms the authenticity of the client. This has to be done in a specific time frame because if the time taken exceeds the specified time limit the request is canceled. The connection is terminated only after the legitimacy of the client is confirmed. This could be implemented in Deep space scenario as well to protect devices against De-authentication attack.

The nodes can be made intelligent by training them to accept de-authentication frames only on certain time stamps or in certain conditions. If anything, out of order occurs the node can be prepared for it.

Wireless Intrusion Prevention System (WIPS) can be used to prevent the de-authentication attacks. In WIPS Over the Air (OTA) prevention technique can be utilized. This technique crafts MAC layer frames. More information on this can be found in [32].

CONCLUSIONS AND FUTURE SCOPE

Security is a prime concern of all communications. In the years to come when manned missions will be sent to celestial bodies like Moon or Mars the communication will form the backbone of such missions. So, it is very important to analyze what kind of threats are possible in such scenarios. In deep space, communication is usually done through wireless networks which makes it vulnerable to a variety of attacks. In this thesis, the implementation of attack is done on two physical nodes communicating in conditions like that of deep space network by using Interplanetary Overlay Network (ION) software distribution on the testbed created. The designed network is tested by implementing a sending operation on nodes where bundles are sent across the network. To test how the network is affected in an event of attack two test cases were used. In the first test case, a time-based de-authentication attack was performed for a short period to see how much delay it induces in the transmission of bundles. In the second test case, a contact window exhaustion attack was performed in which the bundles were being transmitted from one node to another and parallelly the de-authentication attack was launched for a period equal to the contact window of the nodes. When the attack has been launched the nodes get disconnected from the network which results in the delayed transmission of bundles or no transmission of a part of bundles. Space programs are expensive projects with limited resources and these tests have let us understand how costly even a second's delay can prove to be.

In the future, the network can be tested against more malicious attacks that threaten the security of deep space missions. An experimental study can be carried out about the most malicious terrestrial attacks that are also applicable on the DSN (Deep Space Network). To protect devices from such attacks effective security mechanisms need to be developed that will increase the security of the device without trading off the performance of the device. The prevention schemes suggested for a de-authentication attack on Earth can be tested for DSN as well. Several other parameters can be considered while evaluating the effect of an attack like varying link capacities, the processing power of a node, storage capacities of nodes, detection mechanisms, etc. IPN is still an evolving field that will present itself with a lot of vulnerabilities.

REFERENCES

[1] "Deep Space Network" NASA. Accessed on: Jan 5, 2019. [Online]. Available: https://www.nasa.gov/directorates/heo/scan/services/networks/dsn

[2] "Near Earth Network" NASA. Accessed on: Jan 5, 2019. [Online]. Available: https://www.nasa.gov/directorates/heo/scan/services/networks/nen

[3] "Deep Space Communication" Jet Propulsion Labs. Accessed on: Mar 15, 2019. [Online]. Available: https://scienceandtechnology.jpl.nasa.gov/research/research-topics-list/communications-computing-software/deep-space-communications

[4] "Interplanetary Internet" Wikipedia. Accessed on: Oct 15, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Interplanetary_Internet

[5] Ian F. Akyildiz, OOzgur B. Akan, Chao Chen, Jian Fang, Weilian Su "InterPlaNetary Internet: state-of-the-art and research challenges". Computer Networks, 43(2), 75-112, Elsevier,2003.

[6] "DSN Now" NASA JPL. Accessed on: Mar 20, 2019. [Online]. Available: https://eyes.nasa.gov/dsn/dsn.html

[7] Andrew S. Tanenbaum, "Computer Networks", 4thed., Prentice Hall, 2003

[8] "UK-DMC" Wikipedia. Accessed on: Apr 20, 2019. [Online]. Available: https://en.wikipedia.org/wiki/UK-DMC

[9] "UK-DMC satellite first to transfer sensor data from space using 'bundle' protocol" Surrey Satellite Technology Limited Archive: https://web.archive.org/web/20120426220355/http://www.sstl.co.uk/news_and_events/latest_news?story=1254

[10] "NASA Tests First Deep-Space Internet" NASA. Accessed on: Apr 21, 2019. [Online]. Available: https://www.nasa.gov/topics/technology/features/internet-20081118.html

[11] Kevin Fall "A Delay-Tolerant Network Architecture for Challenged Internets". SIGCOMM'03, August 25-29, 2003.

[12] V.Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, et al, "RFC 4838, Delay-Tolerant Networking Architecture," IRTF DTN Research Group, April 2007.

[13] S. Burleigh, M. Ramadas, et al., "RFC 5325, Licklider Transmission Protocol Motivation," IRTF DTN Research Group, September 2008.

[14] K. Scott, S. Burleigh, et al., "RFC 5050, Bundle Protocol Specification," IRTF DTN Research Group, November 2007.

[15] S. Symington, S. Farrell, H. Weiss, P. Lovell, et al, "RFC 6257 Bundle Security Protocol Specification," IRTF DTN Research Group, May 2011.

[16] S. Farrell, M. Ramadas, S. Burleigh, et al, "RFC 5327 Licklider Transmission Protocol - Security Extensions" IRTF DTN Research Group, September 2008.

[17] Joyeeta Mukherjee and Byrav Ramamurthy "Communication Technologies and Architectures for Space Network and Interplanetary Internet" IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 15, NO. 2, SECOND QUARTER 2013

[18] Mohanchur Sarkar, K.K.Shukla, K.S.Dasgupta "A Survey of Transport Protocols for Deep Space Communication Networks". International Journal of Computer Applications (0975 – 8887) Volume 31– No.8, October 2011.

[19] Sara El Alaoui, Saichand Palusa and Byrav Ramamurthy "The Interplanetary Internet Implemented on the GENI Testbed" Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE, 2015.

[20] "Wi-Fi deauthentication attack" Wikipedia. Accessed on: Oct 23, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Wi-Fi_deauthentication_attack

[21] 5th HLF – Lecture: Vinton Gray Cerf. (Sept. 29, 2017). Accessed: June 3, 2019. [Online Video]. Available: https://www.youtube.com/watch?v=eH0m5DCogxQ

[22] Thi Ngoc Diep Pham and Chai Kiat Yeo "Detecting Colluding Blackhole and Greyhole Attacks in Delay Tolerant Networks" IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 15, NO. 5, MAY 2016

[23] Md Yusuf Sarwar Uddin, Ahmed Khurshid, Hee Dong Jung, Carl Gunter, Matthew Caesar, Tarek Abdelzah "Making DTNs Robust Against Spoofing Attacks with Localized Countermeasures" 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, 2011

[24] Md Yusuf Sarwar Uddin, Ahmed Khurshid, Hee Dong Jung, Carl Gunter, et al, "Denial in DTNs" 2011

[25] Jet Propulsion Labs Interplanetary Overlay Network (ION) Design and Operation. (2011). [Online]. Available: https://github.com/NASAHackTO/ion-dtn/blob/master/ion-open-source/ION.pdf

[26] Interplanetary Overlay Network. (3.6.2). JPL. Accessed: Jan 10, 2019. [Online]. Available: https://sourceforge.net/projects/ion-dtn/

[27] Delay- and Disruption-Tolerant Networks (DTNs) A Tutorial. Forrest Warthman, Warthman Associates. Accessed: Feb 10, 2019. [Online]. Available: http://ipnsig.org/wp-content/uploads/2015/09/DTN_Tutorial_v3.2.pdf

[28] Joyeeta Mukherjee, "Routing over the Interplanetary Internet" M.S. thesis, Dept. Computer Science, University of Nebraska, Lincoln, Nebraska, USA, 2012.

[29] "spacenetworking" Youtube (Jun 21, 2012). Accessed on: Aug 26, 2018. [Online Video]. Available: https://www.youtube.com/watch?v=nWtRTzXJvtI

[30] Sara El Aloui, "Routing Optimization in Interplanetary Networks" M.S. thesis, Dept. Computer Science, University of Nebraska, Lincoln, Nebraska, USA, 2015.

[31] M. Ramadas, S. Burleigh, S. Farrell, et al, "RFC 5326 Licklider Transmission Protocol - Specification" IRTF DTN Research Group, September 2008.

[32] Vartak, A., Ahmad, S., & Gopinath, K. N. (2007, January). An experimental evaluation of over-the-air (ota) wireless intrusion prevention techniques. In *2007 2nd International Conference on Communication Systems Software and Middleware* (pp. 1-7). IEEE.

APPENDIX A

Installing ION software

ION is an open software module provided by JPL. It can be downloaded from https://sourceforge.net/projects/ion-dtn/. ION version 3.6.2 for implementing this experiment. Some pre-requisites were used for successfully implementing ION:

- Linux operating system (14.04, 16.04)
- The Expat XML parser and development libraries.
- Standard GNU and build tools like tar, gzip, etc.

Before installing ION make sure you have Expat parser installed in your systems. It can be downloaded from https://sourceforge.net/projects/expat/. ION allows you to implement a lot of protocols from the DTN protocol stack like :

Table 4 Protocols that are supported by ION

Bundle Protocol (BP)	Licklider Transmission Protocol (LTP)	CCSDS File Delivery Protocol (CFDP)	Asynchronous Message Service (AMS)	Interplanetary Communication Infrastructure (ICI)	Datagram Retransmission (DGR)

The commands utilized for installing ION and Expat XML parser are as follows:

```
1. tar -xvzf expat-2.1.0.orig.tar.gz
2. cd expat-2.1.0
3. ./configure –prefix=/usr
4. make
5. sudo make install
```

Fig 36. Commands for installing Expat XML parser

```
1. tar -xvzf ion-3.6.2.tar.gz
2.  cd ion-3.6.2
3. ./configure
4. make
5. sudo make install
6. sudo ldconfig
```

Fig 37. Commands for installing ION software

Once the installation of all components has completed it is tested out by creating a configuration file that contains information about the network like Contact Plans (CP), Protocol to be used, bytes of data that can be sent, naming schemes, induct and outducts of the network, etc. You can test your machine by using loopback address i.e. try sending a message to yourself by using loopback addresses as source and destination address. After the nodes are correctly configured data can be transmitted across them.

APPENDIX B

End to End Data transmission of data from Spacecraft to Mission Control Center

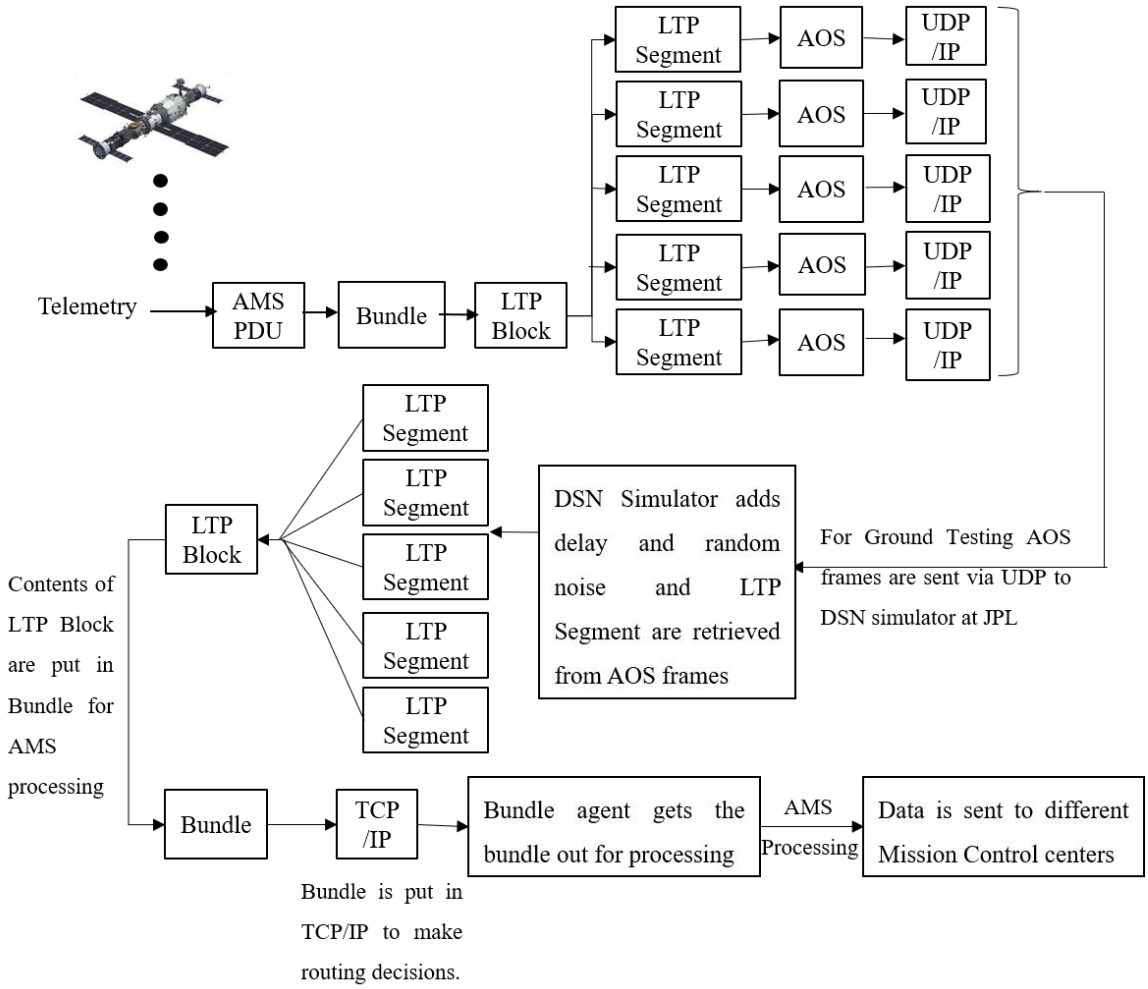

Fig 38. Data transmission from Spacecraft to Mission Control Center (MCC)

The figure above demonstrates how the data is transmitted from a Spacecraft to the MCC i.e. end to end communication. In this particular scenario, the data is being transmitted by spacecraft to MCC, not the other way around. The process starts when the spacecraft starts sending the telemetry, from here on encapsulation and decapsulation processes start. The data sent by the spacecraft is sent via Asynchronous Message Service (AMS) in a bundle using LTP for reliability. If the file is bigger than specified limits then the LTP Block is divided up into segments forming LTP Segments. For ground station testing the LTP

segments are then put into Advanced Orbiting System (AOS) telemetry frames for transmission via User Datagram Protocol (UDP) to Deep Space Network (DSN) simulator at JPL. As the data is received by the DSN simulator the AOS frames are decapsulated and LTP segments are extracted. Alongside this, the DSN simulator adds random noise and One Way Light Time (OWLT) delay. As the LTP Segments are being amalgamated to form an LTP Block, in case there is a lost LTP segment encountered the request for retransmission is sent. The bundle acquired from the LTP Block is now sent for AMS processing by Bundle Agent using TCP/IP so that the routing decisions can be made. The AMS processor now yields the results to the specific users registered for that service.

APPENDIX C

Codes Used in Thesis

C.1 Code to find SSID, Channel number and BSSID of an access point

```
import socket
import struct
import shelve
import os
import sys
import traceback
ch = raw_input("Press 'Y' to know previous result ")
print "USE only Ctrl+c to exit "
try :
        if ch.lower() == 'y':
                s = shelve.open("wireless_data.dat")
                print "Seq", "\tBSSID\t\t", "\tChannel", "SSID"
                keys= s.keys()
                list1 = []
                for each in keys:
                        list1.append(int(each))
                list1.sort()

                for key in list1:
                        key = str(key)
                        print key,"\t",s[key][0],"\t",s[key][1],"\t",s[key][2]
                s.close()
                raw_input("Press any key to continue ")
except Exception as e :
        print e
        raw_input("Press any key to continue ")


try:
        sniff = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, 3)
        sniff.bind(("mon0", 0x0003))

except Exception as e :
        print e
ap_list =[]
print "Seq", "\tBSSID\t", "\t\tChannel", "SSID"

s = shelve.open("wireless_data.dat","n")
try:
        while True :
```

```
                fm1 = sniff.recvfrom(6000)
                fm= fm1[0]
                radio_tap_lenght = ord(fm[2])
                #print radio_tap_lenght
                if fm[radio_tap_lenght] == "\x80" :
                        source_addr =
fm[radio_tap_lenght+4+6:radio_tap_lenght+4+6+6]
                        #print source_addr
                        if source_addr not in ap_list:
                                ap_list.append(source_addr)
                                byte_upto_ssid = radio_tap_lenght+4+6+6+6+2+12+1
                                a = ord(fm[byte_upto_ssid])
                                list_val = []
                                #print a
                                bssid = ':'.join('%02x' % ord(b) for b in source_addr)
                                #bssid = fm[36:42].encode('hex')
                                s_rate_length = ord(fm[byte_upto_ssid+1 +a+1])
                                channel = ord(fm[byte_upto_ssid+1
+a+1+s_rate_length+3])
                                ssid = fm[byte_upto_ssid+1:byte_upto_ssid+1 +a]
                                print len(ap_list),"\t",bssid,"\t",channel,"\t",ssid
                                list_val.append(bssid)
                                list_val.append(channel)
                                list_val.append(ssid)
                                seq = str(len(ap_list))

                                s[seq]=list_val
except KeyboardInterrupt:
        s.close()
        sys.exit()

except Exception as e :
        traceback.print_exc()
        print e
```

C.2 Code for De-authentication attack

```
import socket
import struct
import shelve
import os
import sys
import traceback
ch = raw_input("Press 'Y' to know previous result ")
print "USE only Ctrl+c to exit "
try :
        if ch.lower() == 'y':
                s = shelve.open("wireless_data.dat")
                print "Seq", "\tBSSID\t\t", "\tChannel", "SSID"
                keys= s.keys()
                list1 = []
                for each in keys:
                        list1.append(int(each))
                list1.sort()

                for key in list1:
                        key = str(key)
                        print key,"\t",s[key][0],"\t",s[key][1],"\t",s[key][2]
                s.close()
                raw_input("Press any key to continue ")
except Exception as e :
        print e
        raw_input("Press any key to continue ")


try:
        sniff = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, 3)
        sniff.bind(("mon0", 0x0003))

except Exception as e :
        print e
ap_list =[]
print "Seq", "\tBSSID\t", "\t\tChannel", "SSID"

s = shelve.open("wireless_data.dat","n")
try:
        while True :
                fm1 = sniff.recvfrom(6000)
                fm= fm1[0]
                radio_tap_lenght = ord(fm[2])
                #print radio_tap_lenght
```

```
                if fm[radio_tap_lenght] == "\x80" :
                        source_addr =
                        fm[radio_tap_lenght+4+6:radio_tap_lenght+4+6+6]
                        #print source_addr
                        if source_addr not in ap_list:
                                ap_list.append(source_addr)
                                byte_upto_ssid = radio_tap_lenght+4+6+6+6+2+12+1
                                a = ord(fm[byte_upto_ssid])
                                list_val = []
                                #print a
                                bssid = ':'.join('%02x' % ord(b) for b in source_addr)
                                #bssid = fm[36:42].encode('hex')
                                s_rate_length = ord(fm[byte_upto_ssid+1 +a+1])
                                channel =
                                ord(fm[byte_upto_ssid+1+a+1+s_rate_length+3])
                                ssid = fm[byte_upto_ssid+1:byte_upto_ssid+1 +a]
                                print len(ap_list),"\t",bssid,"\t",channel,"\t",ssid
                                list_val.append(bssid)
                                list_val.append(channel)
                                list_val.append(ssid)
                                seq = str(len(ap_list))

                                s[seq]=list_val
except KeyboardInterrupt:
        s.close()
        sys.exit()

except Exception as e :
        traceback.print_exc()
        print e
```

C.3 Code for Detecting De-authentication attack

```
import socket
import Queue
from threading import Thread
from collections import Counter

q1 = Queue.Queue()
co = Counter()

try:
        sniff = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, 3)
        sniff.bind(("mon0", 0x0003))
except Exception as e :
        print e

def ids():
        global q1
        while True :
                fm1 = sniff.recvfrom(6000)
                fm= fm1[0]
                radio_tap_lenght = ord(fm[2])
                if ord(fm[radio_tap_lenght]) == 192:
                        bssid1 = fm[radio_tap_lenght+4+6+6
:radio_tap_lenght+4+6+6+6]
                        bssid = ':'.join('%02x' % ord(b) for b in bssid1)
                        q1.put(bssid)
                        #print "Deauth attack on", bssid


def insert_frame():
        global q1
        while True:
                mac=q1.get()
                list1 = [mac]
                co.update(list1)
                print dict(co)

i = Thread(target=ids)
f = Thread(target=insert_frame)
i.start()
f.start()
```

C.4 Code to find hidden SSID

```
from scapy.all import *
interface ='mon0'
probe_req = []

BSSID = raw_input("Enter the MAC of AP ")
def probesniff(fm):
        if fm.haslayer(Dot11ProbeReq):
                if BSSID == fm.addr2 :
                        print fm.info

sniff(iface= interface,prn=probesniff)
```

C.5 Code to find the Clients of a specific Access Point

```
from scapy.all import *
interface ='mon0'
probe_req = []
ap_name = raw_input("Please enter the AP name ")
def probesniff(fm):
        if fm.haslayer(Dot11ProbeReq):
                client_name = fm.info
                if client_name == ap_name :
                        if fm.addr2 not in probe_req:
                                print "New Probe Request: ", client_name
                                print "MAC ", fm.addr2
                                probe_req.append(fm.addr2)

sniff(iface= interface,prn=probesniff)
```