

The background of the slide is a composite image. The top half features a large, white, parabolic satellite dish antenna, likely from a ground station, with its complex support structure visible. The bottom half shows a satellite in orbit above the Earth's cloud-covered surface. The satellite has a cylindrical body and several rectangular solar panel arrays extending from it. A large, dark, diagonal shape, possibly a wing or a part of the presentation design, separates the ground station from the satellite.

Satellite Anomaly Inject & Detection(SAID) Testbed

Powered by CSULA & Aerospace Corporation
Advisor: Zilong Ye

December 3, 2021

CONTENTS

Project and Team Introduction

0

1

OSK/COSMOS Overview

Ground and Flight System walk through

0

2

Current Work & Future Goals

Single Bit Error/DDos resolutions

0

3

Current Milestones

Accomplishments

0

4

Challenges & Questions

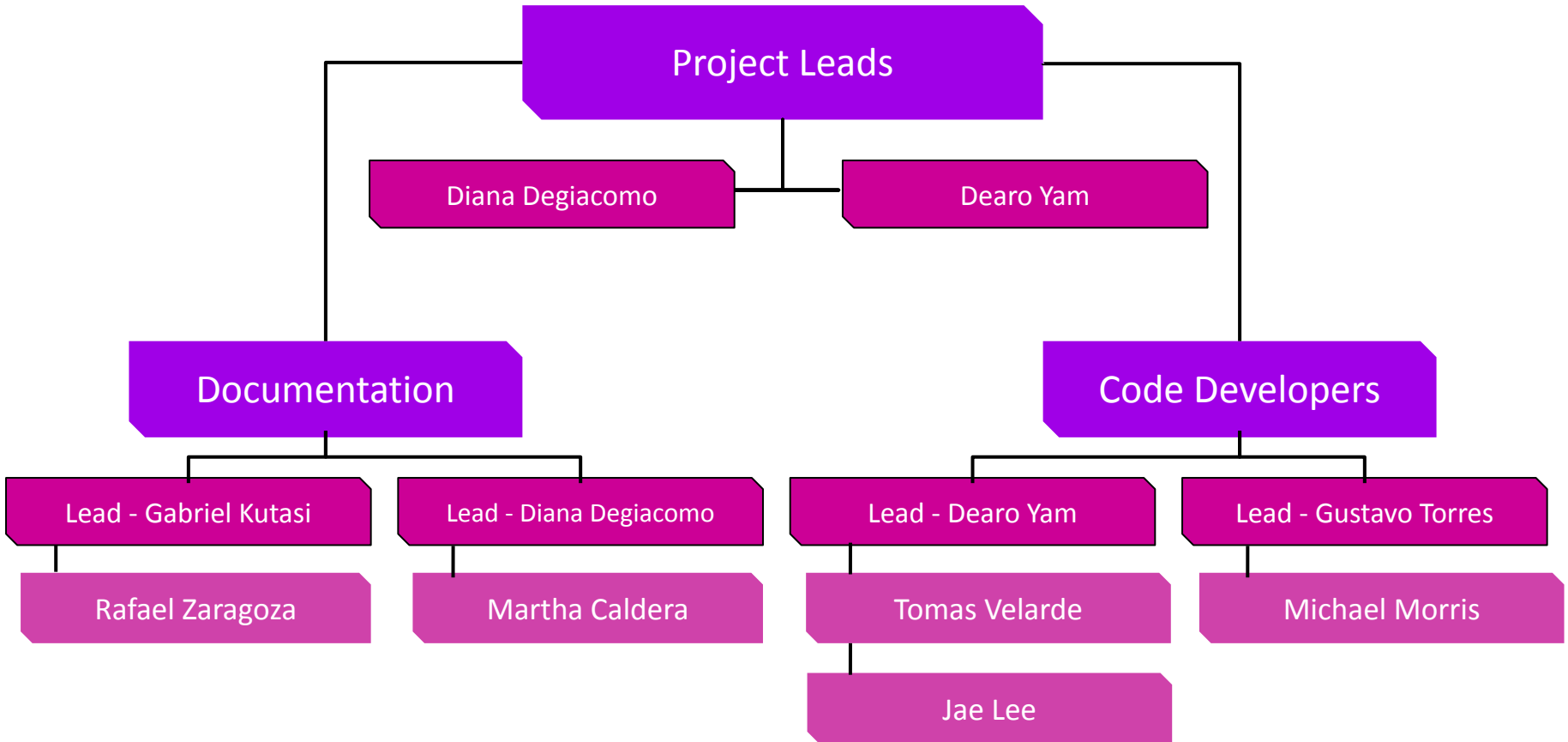
Issues in Development and Q&A

0

5



Organizational Chart - Diana



Welcome Agenda - Diana



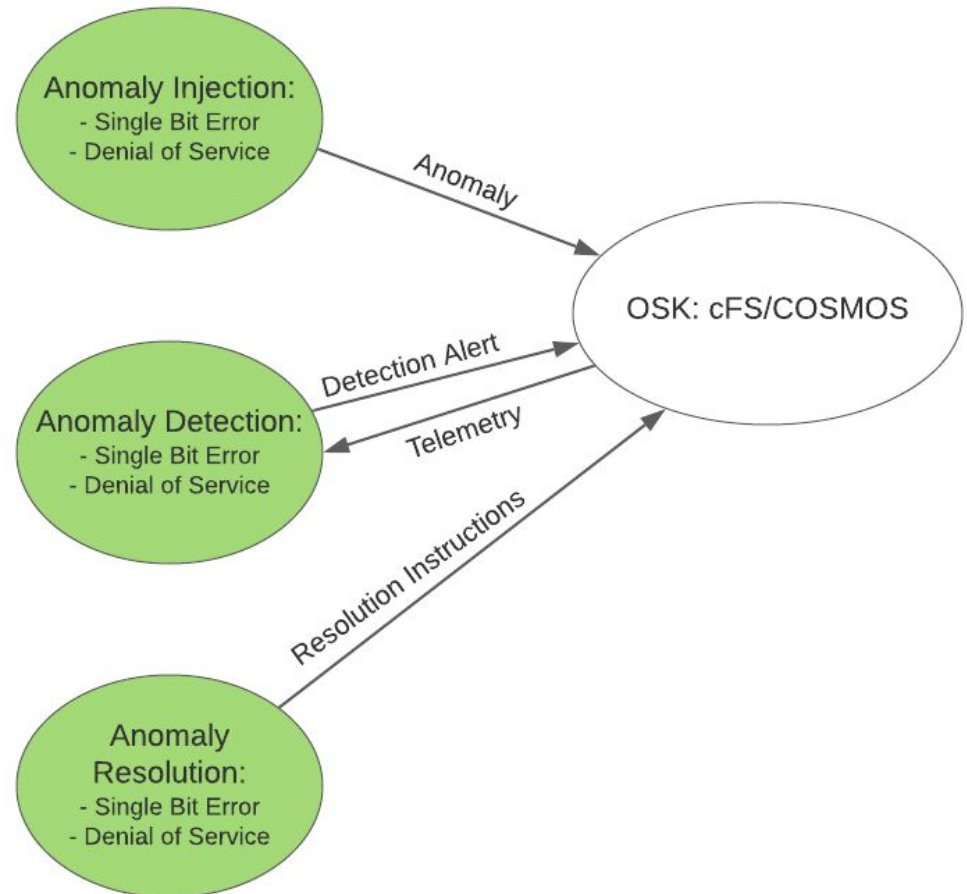
- Project Introduction - Diana Degiacomo
- OpenSatKit - Tomas Velarde
- COSMOS - Jae Lee
- Current Version of Denial of Service - Dearo Yam
- Current Version of Single Bit Error - Gabriel Kutasi
- Future Work for Denial of Service - Martha Caldera
- Future Work for Single Bit Error - Michael Morris
- Milestones - Rafael Zaragoza
- Challenges - Gustavo Torres
- Questions - Gustavo Torres

Project Introduction - Diana

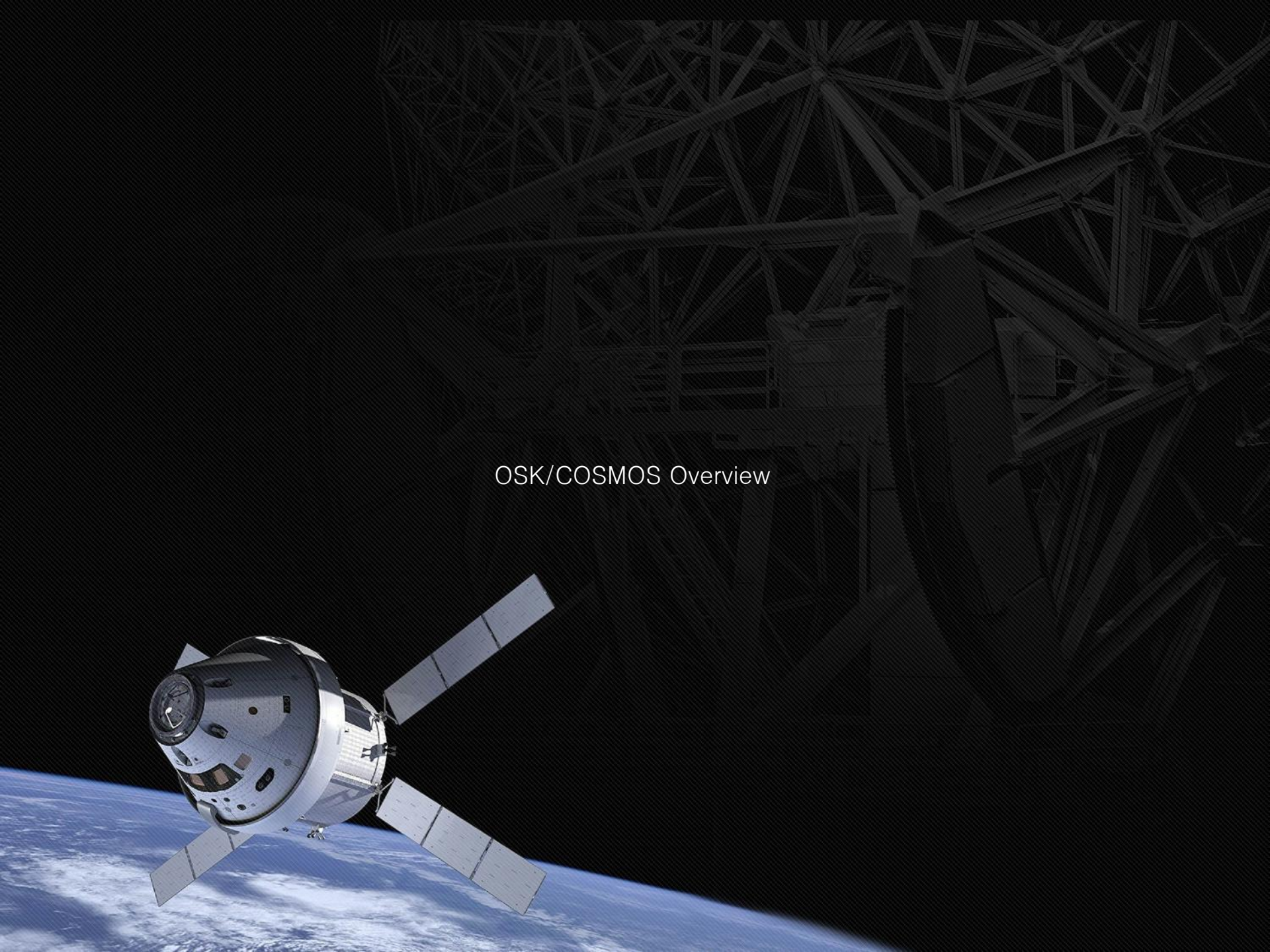


Overall Project Goal: Protect satellites against malicious attacks

- Industry Open Source Software: OSK - A core Flight System (cFS) Platform
 - Build apps to interface with this platform:
 - Inject Anomalies
 - Detect Anomalies
 - Resolve Anomalies
- Anomalies:
 - Single Bit Error
 - Denial of Service



OSK/COSMOS Overview

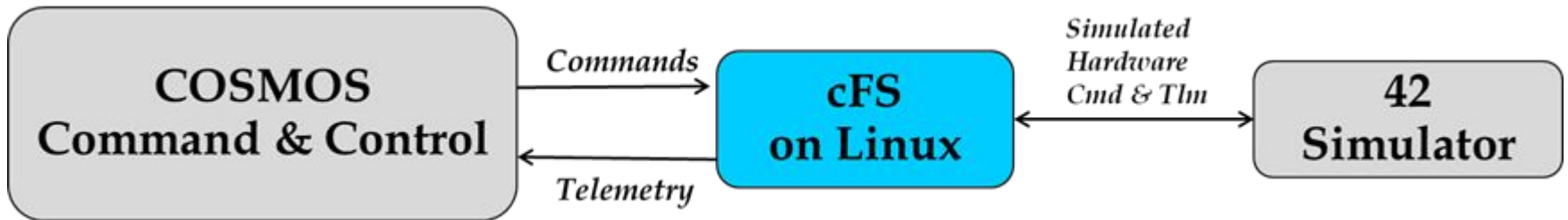


OSK - Tomas Velarde



- **Open Satellite Kit (OSK)** - A satellite simulation environment.
- **Our Team's Goal with OSK-** To use this simulation environment to simulate attacks against satellites and satellite malfunctions to produce effective countermeasures and remedies.

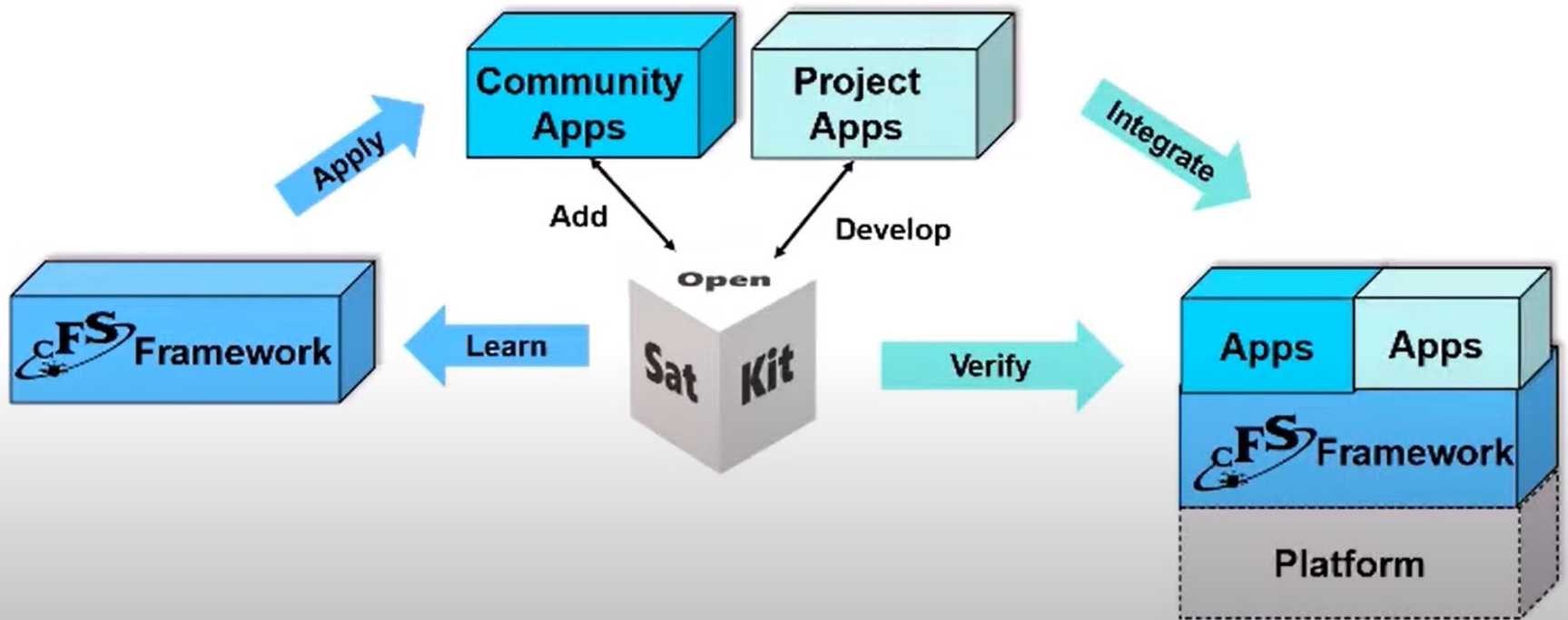
The 3 main components of Open Satellite Kit:



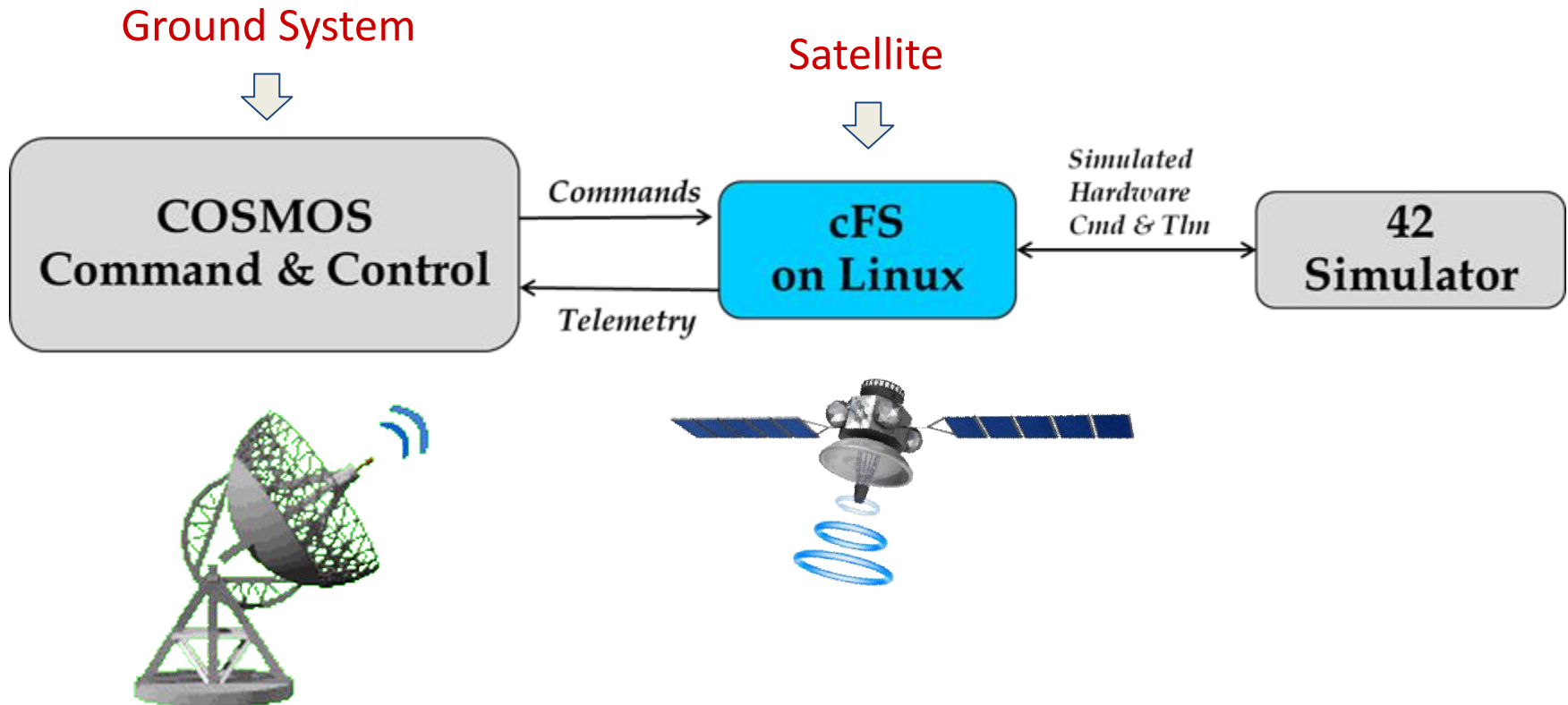
OSK - Tomas Velarde



- Purpose of OSK

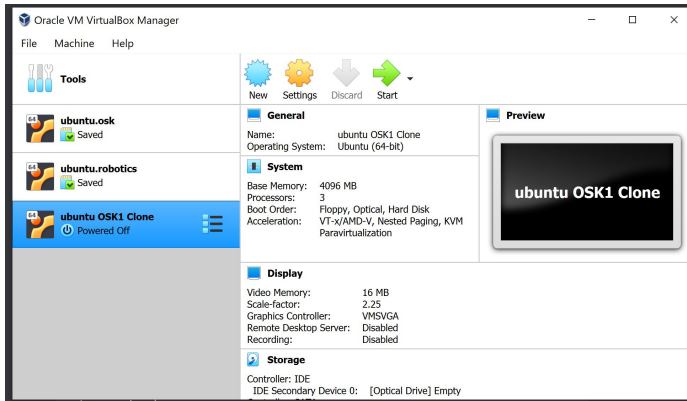


Open Satellite Kit Composition: Three open source tools



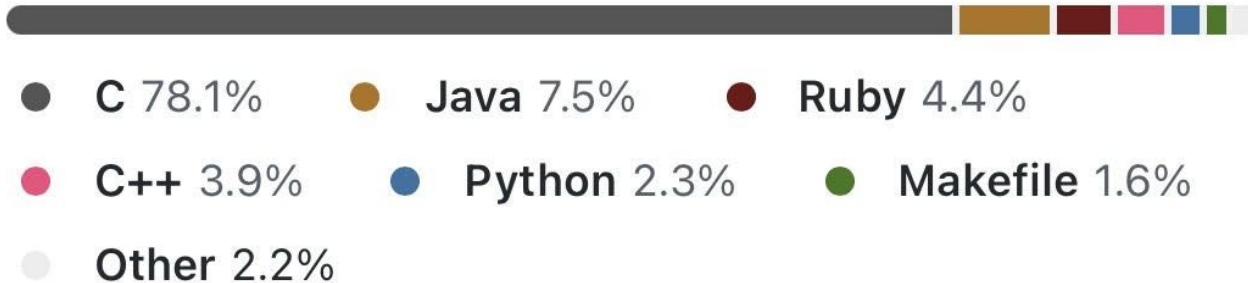


- **Open Satellite Kit runs on Ubuntu Linux**
 - Have installed Oracle VM VirtualBox, a virtual machine set up with Ubuntu Linux to run



- **Programming languages used in Open Satellite Kit:**

Languages



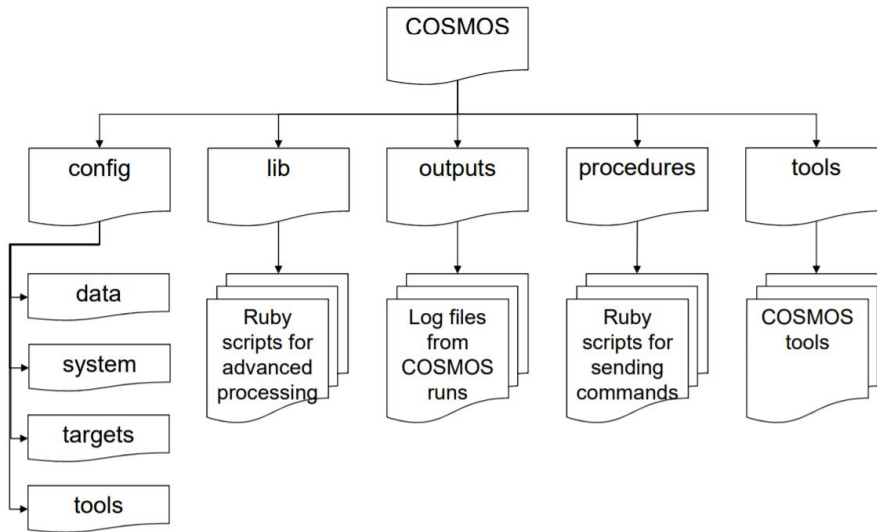
Cosmos - Jae Lee



- *What is COSMOS?*
 - A command and control platform
 - Open-source ground system
 - The primary OSK user interface
- *What is it used for?*
 - To communicate with the satellite
 - Monitor the performance and health
 - Display the data
- *What is it used with?*
 - Test equipment
 - Development boards
 - Satellites



Navigating COSMOS - Jae Lee



- Target
 - Destination for commands
- Tools
 - The Launcher screen consists of tools



COSMOS Tool - Jae Lee



- *Command & Telemetry server tool*
 - Overview of all the connections in the system

cFS Command and Telemetry Server

File Edit Help

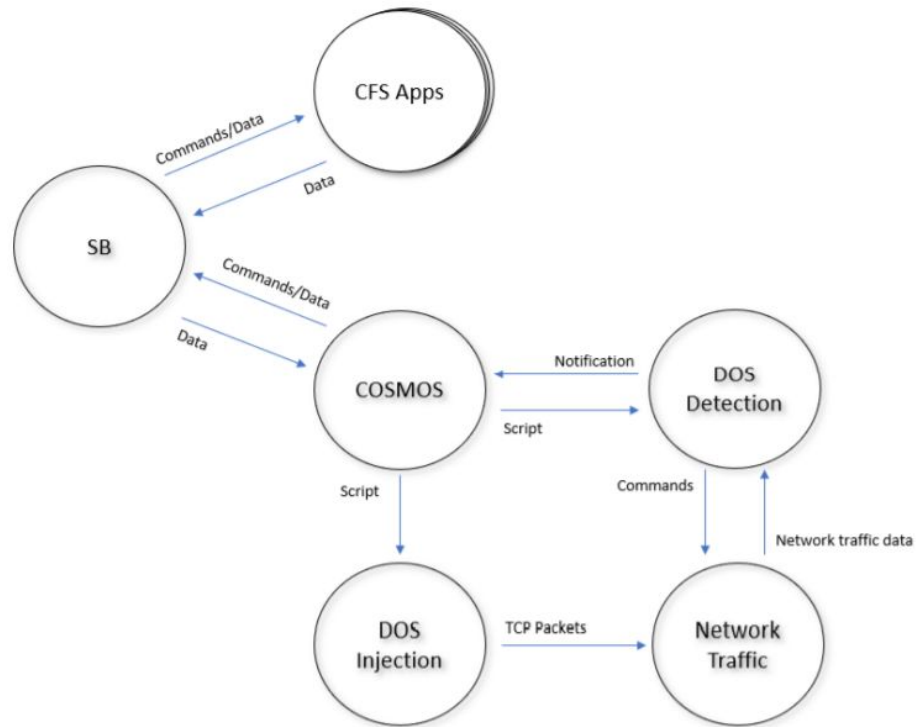
Interfaces Targets Cmd Packets Tlm Packets Routers Logging Status

Interface	Connect/Disconnect	Connected?	Clients	Tx Q Size	Rx Q Size	Bytes Tx	Bytes Rx	Cmd Pkts	Tlm Pkts
COSMOSINT	Disconnect	true	0	0	0	0	0	0	0
CFDP_INT	Disconnect	true	0	0	0	0	123200	0	275
LOCAL_CFS_INT	Disconnect	true	0	0	0	0	461170	0	4164

2021/09/12 12:37:05.425 INFO: Starting connection maintenance for PREIDENTIFIED_CMD_ROUTER
2021/09/12 12:37:05.425 INFO: Connecting to PREIDENTIFIED_CMD_ROUTER...
2021/09/12 12:37:05.427 INFO: Starting packet reading for PREIDENTIFIED_ROUTER
2021/09/12 12:37:05.427 INFO: Connecting to PREIDENTIFIED_ROUTER...
2021/09/12 12:37:05.434 INFO: PREIDENTIFIED_ROUTER Connection Success
2021/09/12 12:37:05.434 INFO: PREIDENTIFIED_CMD_ROUTER Connection Success

Using COSMOS - Jae Lee

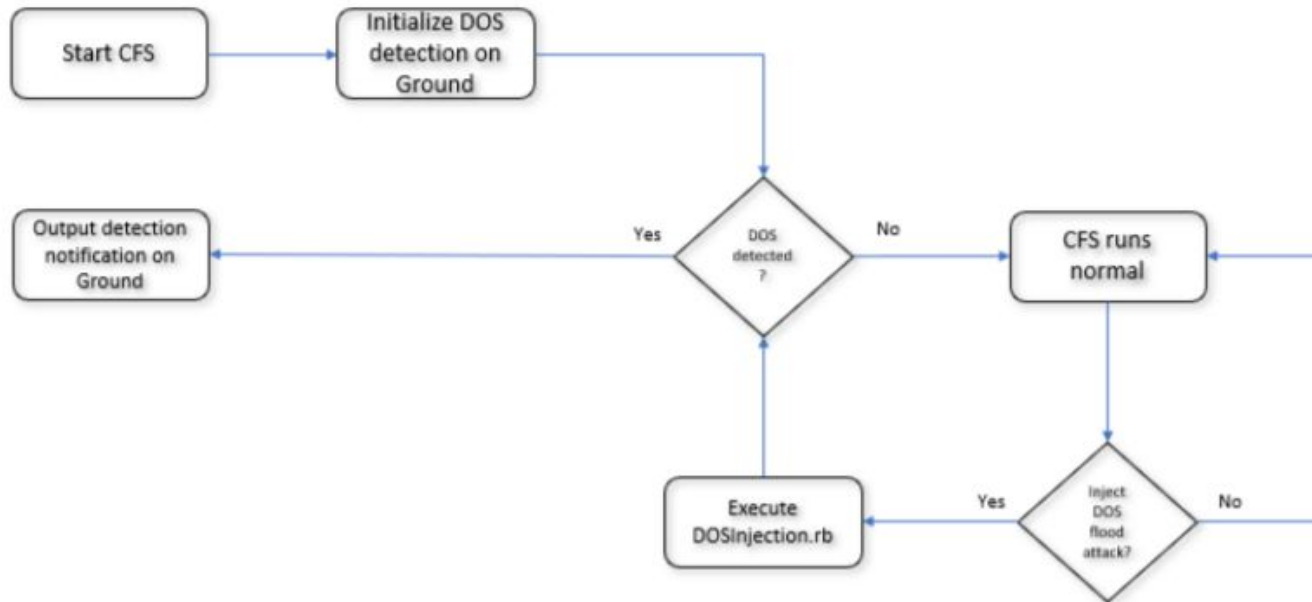
- *Denial of Service (DOS) Injection*



Using COSMOS (cont.) - Jae Lee



- *Denial of Service (DOS) Detection*



Current Work & Future Goals

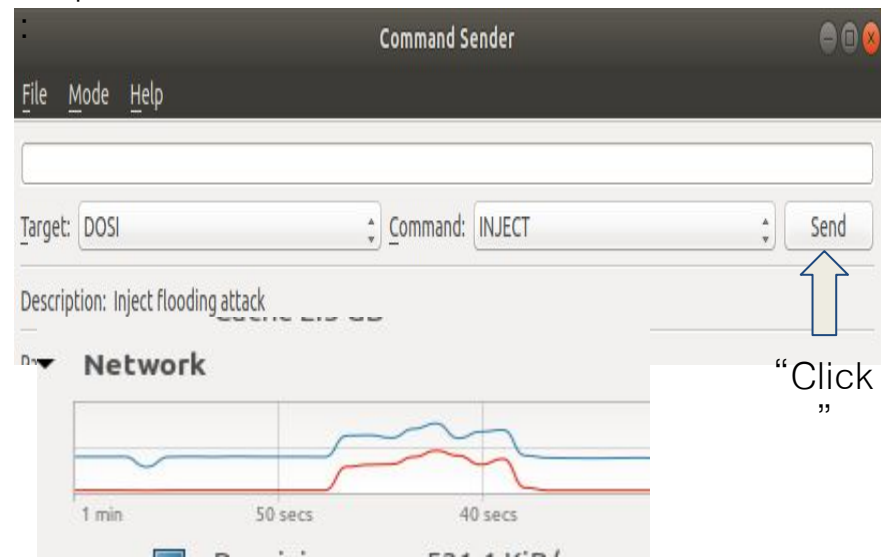


Current Model Of DDOS - Dearo

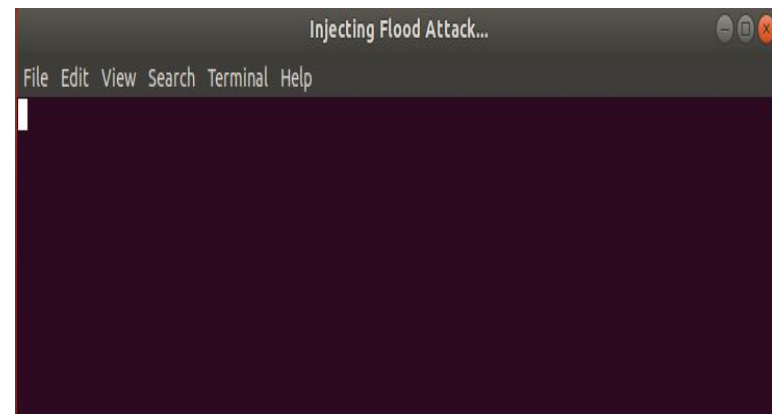
- *Injection*
 - *injects a syn flood attack*
 - *happens for 8 seconds*

```
char command1[] = "gnome-terminal --title='Injecting Flood Attack...' -e 'sudo timeout 8s netwox 76 -i '";  
char command2[] = " -p 23 -s raw' >/dev/null 2>&1 &";  
FILE *ip = popen("hostname -I", "r");  
fscanf(ip, "%s", ipString);  
snprintf(command, 150, "%s%s%s", command1, ipString, command2);  
//printf("%s", command);  
system(command);
```

Step1 ● *Demonstration*



Final:



Current Model Of DDOS (cont.) - Dearo



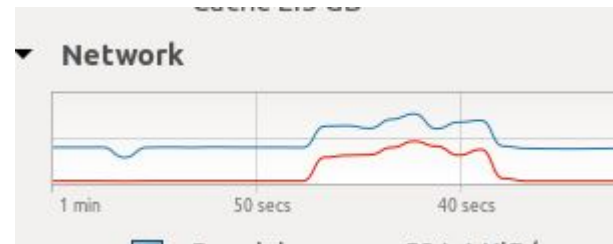
- *Before Attack*

```
Tcp:
  11 active connection openings
  1 passive connection openings
  4 failed connection attempts
  0 connection resets received
  5 connections established
  848 segments received
  797 segments sent out
  4 segments retransmitted
  0 bad segments received
  7 resets sent
```

- *After Attack*

```
Tcp:
  13 active connection openings
  2 passive connection openings
  4 failed connection attempts
  1 connection resets received
  4 connections established
  185135 segments received
  117371 segments sent out
  4 segments retransmitted
  0 bad segments received
  115728 resets sent
```

- During Attack



Current Model Of DDOS (cont.) - Dearo



- *Detection Of DOS*

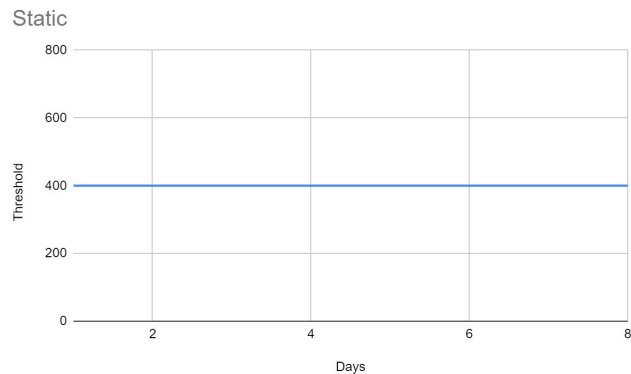
```
file1 = fopen("/sys/class/net/enp0s3/statistics/rx_bytes", "r");  
fscanf(file1, "%d\n", &initByte);
```

```
sleep(1);
```

```
file2 = fopen("/sys/class/net/enp0s3/statistics/rx_bytes", "r");  
fscanf(file2, "%d\n", &byte);
```

```
rate = (byte-initByte)/1000.0;  
//printf("%d : %d = %f\n", byte, initByte, rate);  
//printf("%f\n", rate);
```

```
if(rate > 400.0) {  
    CFE_EVS_SendEvent (DOSD_DETECT_INF_EID, CFE_EVS_INFORMATION, "Network Flooding Detected\n");  
    detected = true;
```



Single Bit Error - Gabe



- *Single Bit Error Injection*

- *Accessed through Command Sender*
- *Injection command is located within the memory manager (MM)*
- *Utilizing PEEK_MEM command will allow the user to view the memory that will be injected with the single bit error command*
- *SBEI_INJECT will flip a bit in the memory, changing the data stored*

```
Terminal
File Edit View Terminal Tabs Help
sch_tbl.json
EVS Port1 42/1/KIT SCH 101: KIT_SCH Initialized. Version 1.1.0
1980-012-14:03:20.30907 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30911 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE TIME 21: Stop FLYWHEEL
EVS Port1 42/1/SC 73: RTS Number 001 Started
EVS Port1 42/1/KIT TO 126: Telemetry output enabled for IP 127.0.0.1
EVS Port1 42/1/KIT SCH 136: Multiple slots processed: slot = 1, count = 2
EVS Port1 42/1/SC 86: RTS 001 Execution Completed
EVS Port1 42/1/KIT SCH 136: Multiple slots processed: slot = 1, count = 2
EVS Port1 42/1/CFE_SB 25: Pipe Overflow,MsgId 0x9d1,pipe KIT_TO_PKT_PIPE,sender
F42
EVS Port1 42/1/CFE_SB 25: Pipe Overflow,MsgId 0x882,pipe KIT_TO_PKT_PIPE,sender
ISIM
EVS Port1 42/1/KIT SCH 136: Multiple slots processed: slot = 1, count = 2
EVS Port1 42/1/KIT SCH 137: Slots skipped: slot = 2, count = 3
EVS Port1 42/1/KIT SCH 136: Multiple slots processed: slot = 1, count = 2
EVS Port1 42/1/KIT SCH 134: Major Frame Sync too noisy (Slot 0). Disabling synch
ronization.
EVS Port1 42/1/MM 7: Peek Command: Addr = 0xF670B0C0 Size = 8 bits Data = 0x00
EVS Port1 42/1/MM 60: SBEI COMMAND: Bit Flipped = 3 Data = 8
```


Single Bit Error - Gabe



- *Single Bit Error Injection*

- *The bits of the application's data is changed when the command is called*
- *The changed data isn't stored to cosmos or cfs*

Command Sender

File Mode Help

Target: MM Command: SBEI_INJECT Send

Description: Inject Single Bit Error

Parameters:

Name	Value or State	Units	Description
CCSDS_STREAMID:	6280		Packet Identification
CCSDS_SEQUENCE:	49152		Packet Sequence Counter
CCSDS_LENGTH:	1		Packet Data Length
CCSDS_CHECKSUM:	0		CCSDS Command Checksum
CCSDS_FUNCODE:	13		Command Function Code

Command History: (Pressing Enter on the line re-executes the command)

```
EVS Port1 42/1/MM 60: SBEI COMMAND: Bit Flipped = 7 Data = 80  
EVS Port1 42/1/MM 60: SBEI COMMAND: Bit Flipped = 6 Data = C0  
EVS Port1 42/1/MM 60: SBEI COMMAND: Bit Flipped = 6 Data = 80  
EVS Port1 42/1/MM 60: SBEI COMMAND: Bit Flipped = 6 Data = C0
```

Future Goals of Denial of Service - Martha



- *Resolution Implementation: Implement machine learning on the ground system for both on **ground** and **onboard** anomalies*
 - *Correction of Denial of Service (DDOS)*
 - *SYN cookies - This is to prevent our server from overloading and crashing*
 - *Machine learning techniques to validate incoming packets*

Future Goals of Single Bit Error - Michael



- *Detection of Single Bit Error (SBE)*
 - *Fully implement Hamming Code*

Adding Parity Bits to 0101 0010 1001 = 0wx1 y010 z010 1001			
Parity #	Check	# of 1's	Parity Bit Value
Parity 1 (w)	_w_1_0_0_0_0_0_1	2	0
Parity 2 (x)	_ _x1_ _10_ _10_ _01	4	0
Parity 4 (y)	_____y010_____1001	3	1
Parity 8 (z)	_____z0101001	3	1
Output: 0001 1010 1011 1001			

Future Goals of Single Bit Error - Michael



- *Correction of Single Bit Error (SBE)*
 - *Hamming code can also correct single bit errors*

0 0 0000	1 0 0001	2 0 0010	3 1 0011
4 1 0100	5 0 0101	6 1 0110	7 0 0111
8 1 1000	9 0 1001	10 1 1010	11 1 1011
12 1 1100	13 0 1101	14 0 1110	15 1 1111

$$\begin{array}{r} 0011 \\ 0100 \\ 0110 \\ 1000 \\ 1010 \\ 1011 \\ 1100 \\ \oplus 1111 \\ \hline 1011 \end{array}$$

Current Milestones

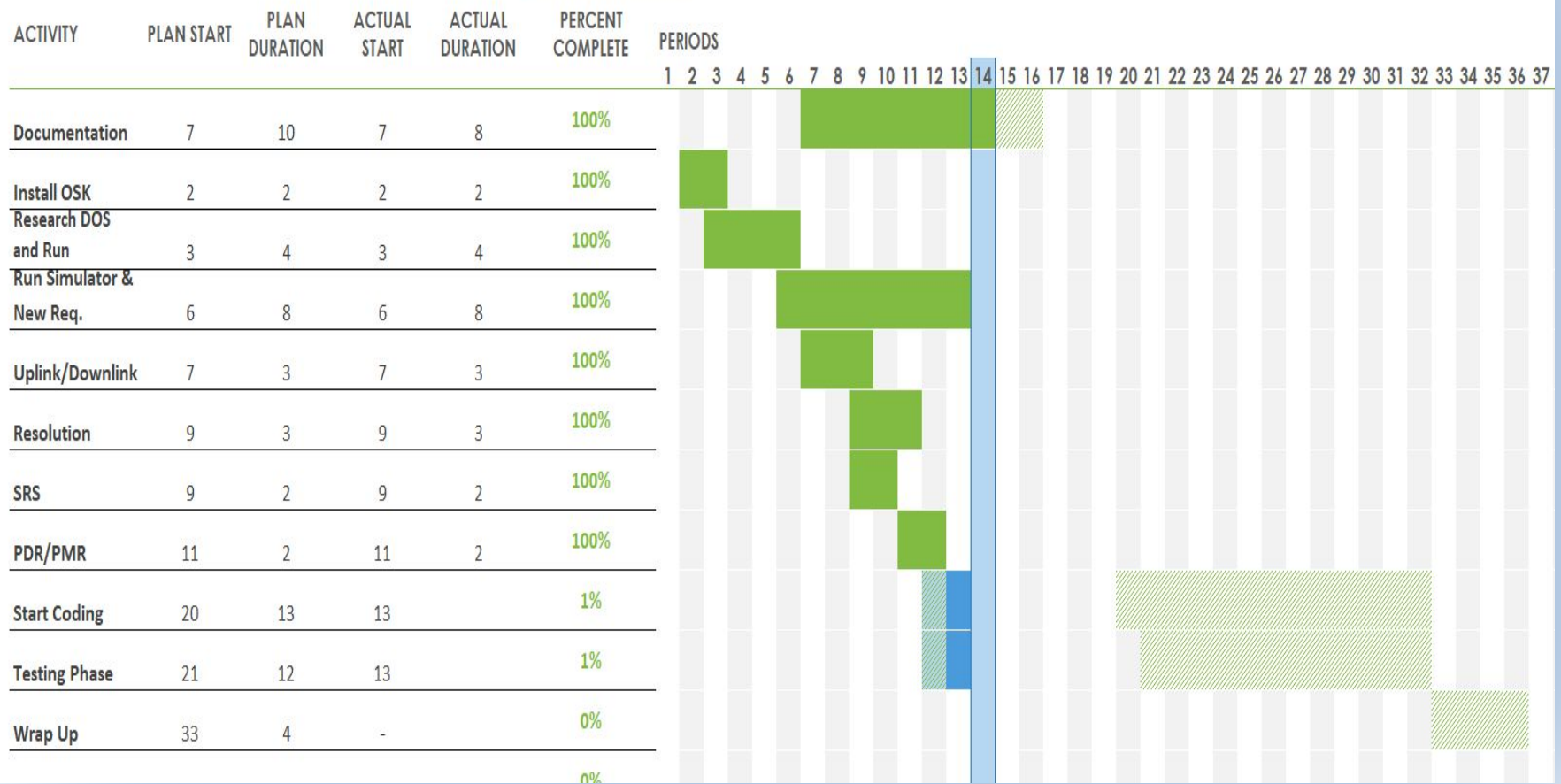




SAID Planner

Period Highlight: 14

Plan Duration
 Actual Start
 % Complete
 Actual (beyond plan)
 % Complete (beyond plan)



Milestones



- *Installed and familiarized with OSK and developed first HELLO WORLD!*
- *Identified the uplink and downlink process*
- *Learned ML amongst the students*
- *Researched datasets to use for ML for our resolutions*
- *Familiarized what is Denial of Service and Single Bit Error*
- *Developed a resolution for DDOS and Single Bit Error*
- *Finalized Documentation such as PDR, PMR, SRS, Timeline*

Challenges & Questions



Challenges - Gustavo



Single Bit Error

- *Not able to read the data from an application twice in one run.*
- *Cannot send a command to satellite while it's already running a command from the same application*
- *Having issues importing the CFE_utils library to another application*

Denial of Service

- *Lack of a proper dataset at the start*
- *Having to learn a new topic (machine learning) to some students*

A screenshot of a satellite command interface. The window has a title bar with 'Command S' and standard OS controls. It features a menu bar with 'File', 'Mode', and 'Help'. Below the menu is a search bar and a 'Send' button. The main area is divided into two panes. The left pane shows a 'Target' dropdown set to 'MM' and a 'Command' dropdown. Below this is a 'Description: Detect Single Bit Error' and a 'Parameters:' section containing a table. The table has columns for 'Name', 'Value or State', 'Units', and 'Command'. The right pane is a scrollable list of commands, with 'SBED_DETECT' and 'SBED_INJECT' highlighted. At the bottom, a 'Command History' section shows a list of commands entered, including 'cmd("MM SBED_DETECT with CCSDS_STREAMID 6280, CCSDS_SEQUENCE 49152, CCSDS_LENGTH 1, CCSDS_CHECKSUM 0, CCSDS_FUNC CODE 13")' and 'cmd("MM SBED_DETECT with CCSDS_STREAMID 6280, CCSDS_SEQUENCE 49152, CCSDS_LENGTH 1, CCSDS_CHECKSUM 0, CCSDS_FUNC CODE 14")'.

- **Gustavo**



Questions and Comments



THANK YOU

