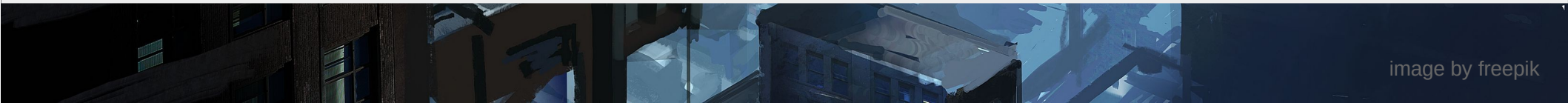
Open Source Ecosystem for Space and Earth Exploration

LIBRECUBE

Artur Scholz
Open Source CubeSat Workshop
9 - 10 December 2021

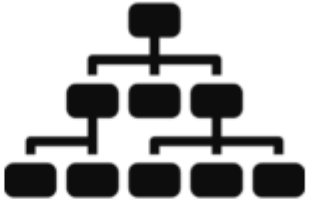


Open Source

Everything we do at LibreCube is made available to the public as free and open source. And we only use free and open source tools – this way, really everyone can get involved!

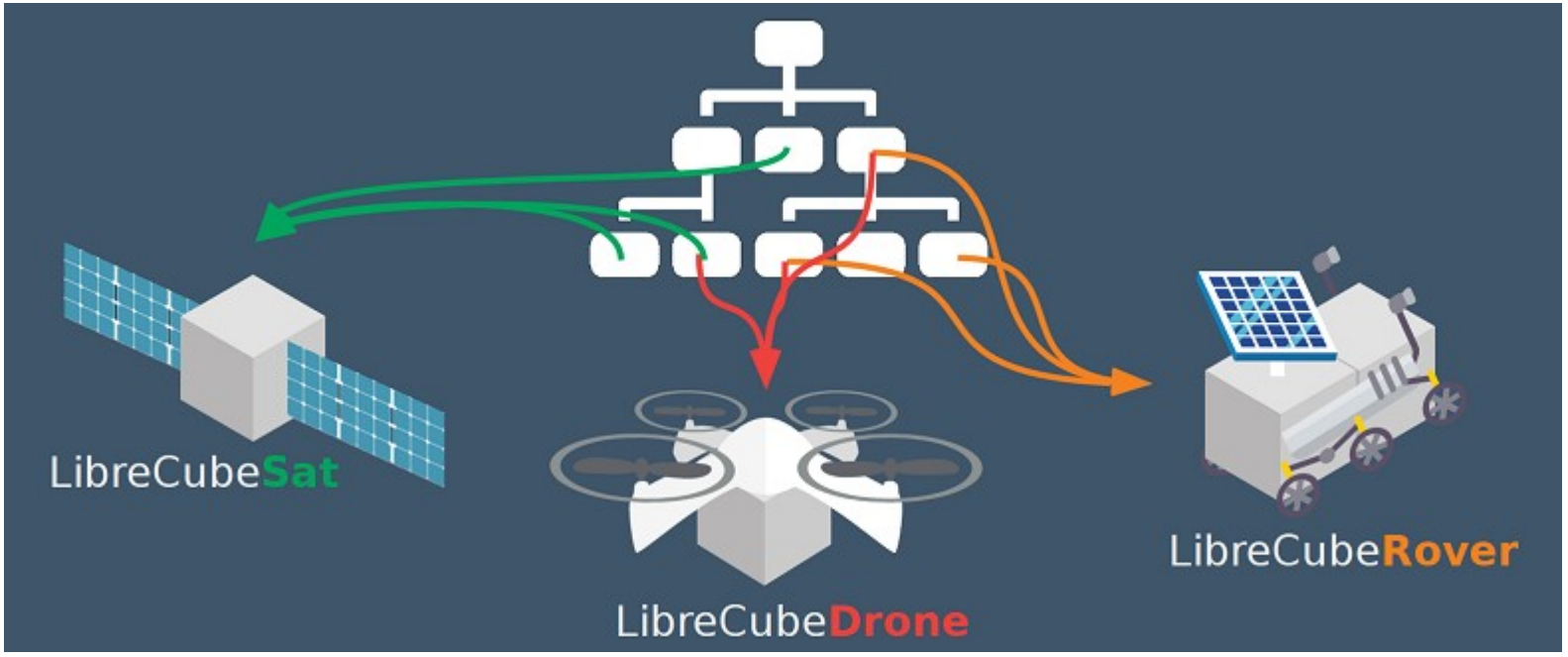
Free and Open Standards

We rely on proven and tested standards for our system designs, with preference to standards from the space domain.

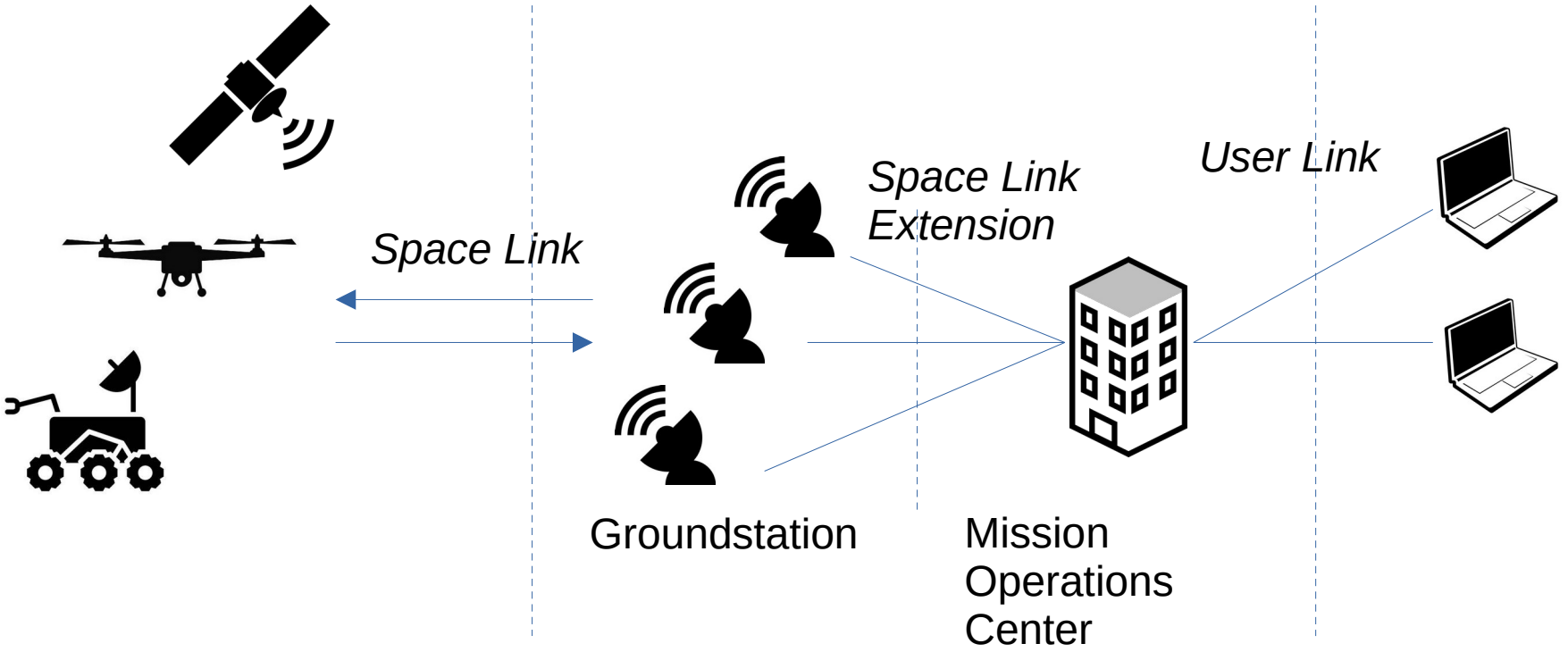


Reference Architecture

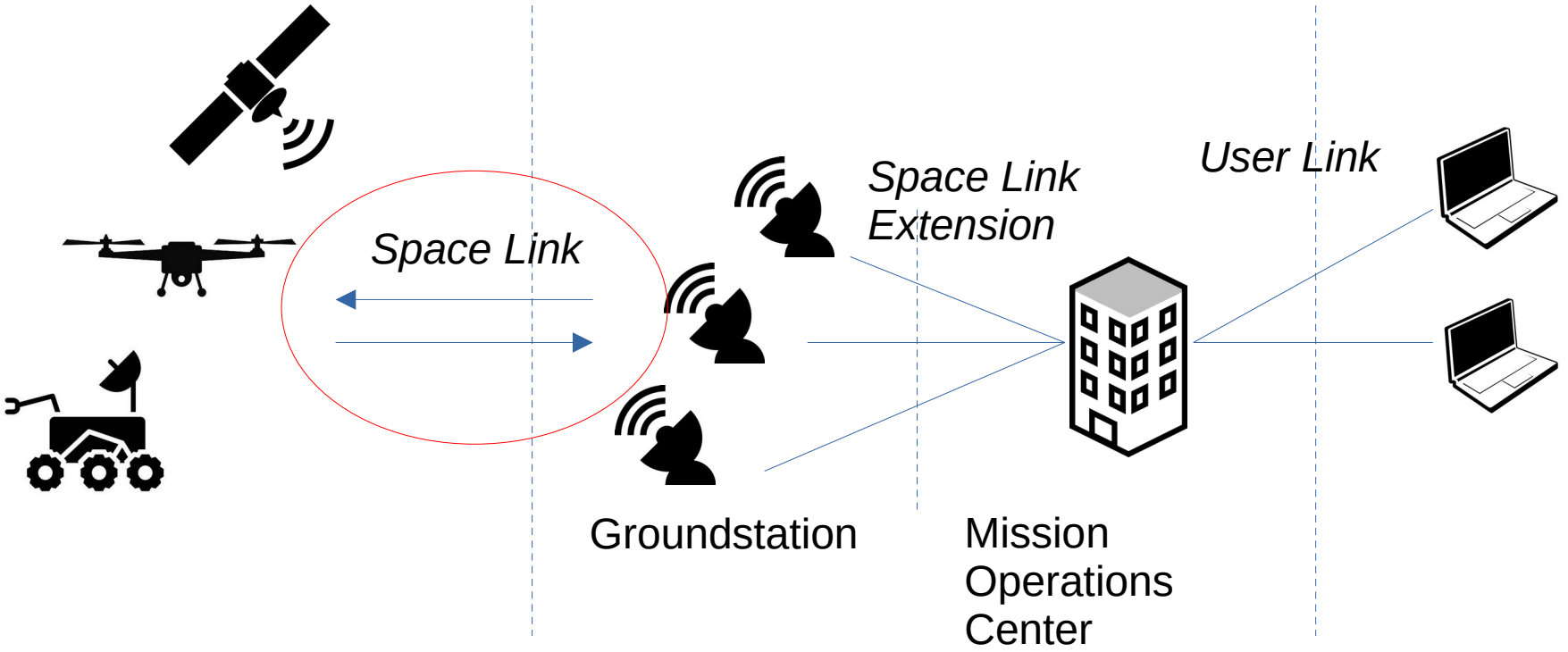
Defining a generic architecture of system of systems that have standardized interfaces makes it possible to combine and reuse elements for various applications.


LibreCubeSat
LibreCubeDrone
LibreCubeRover

Segments


Space Link
Space Link Extension
User Link
Groundstation
Mission Operations Center
Space/Remote Segment
Ground Segment
User Segment

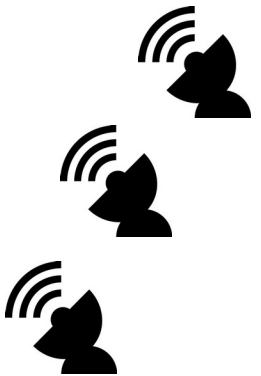
Segments


Space Link
Space Link Extension
User Link
Groundstation
Mission Operations Center
Space/Remote Segment
Ground Segment
User Segment

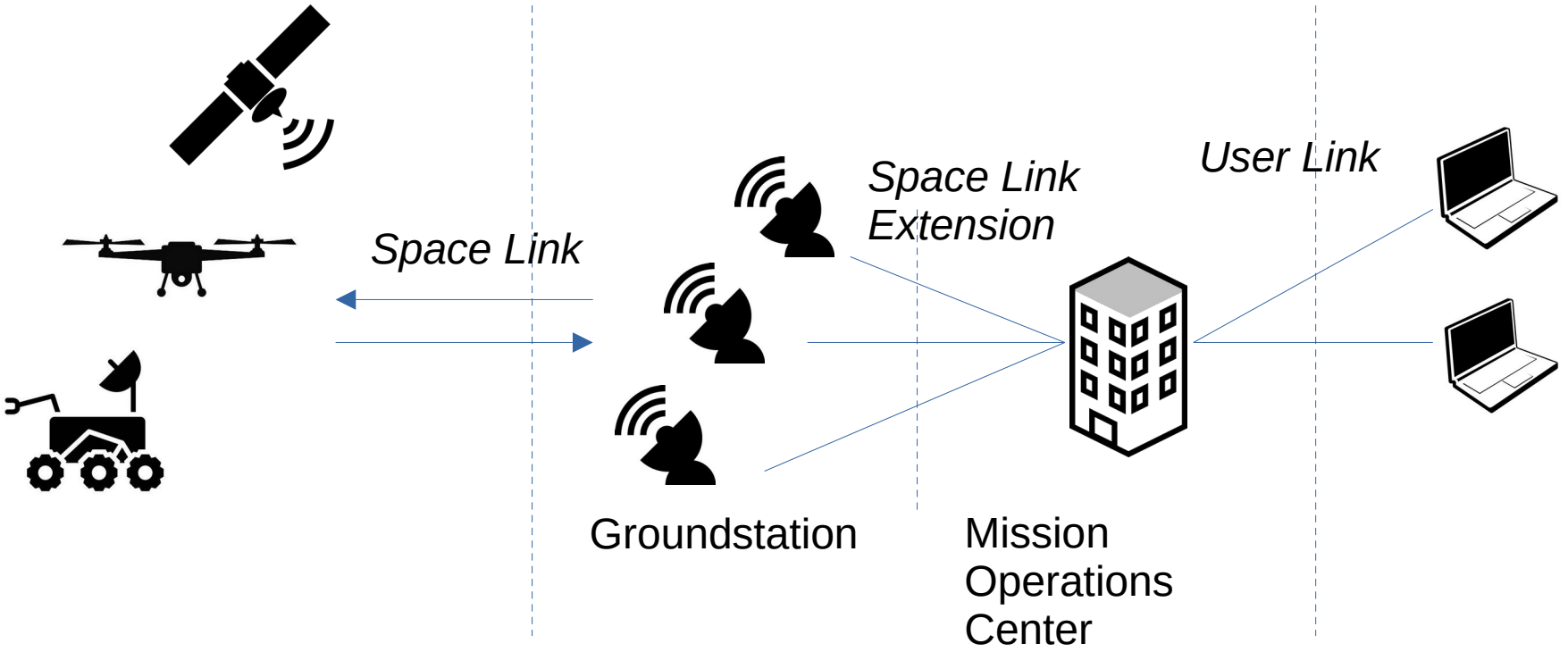
Space Link

Telecommands	
Application Layer	- ECSS PUS Services - CCSDS File Delivery Protocol
Transport Layer	-/-
Network Layer	Space Packet Protocol
Data Link Layer (Protocol)	TC Space Data Link Protocol
Data Link Layer (Sync. And Channel Coding)	TC Sync. And Channel Coding
Physical Layer	RF and Modulation

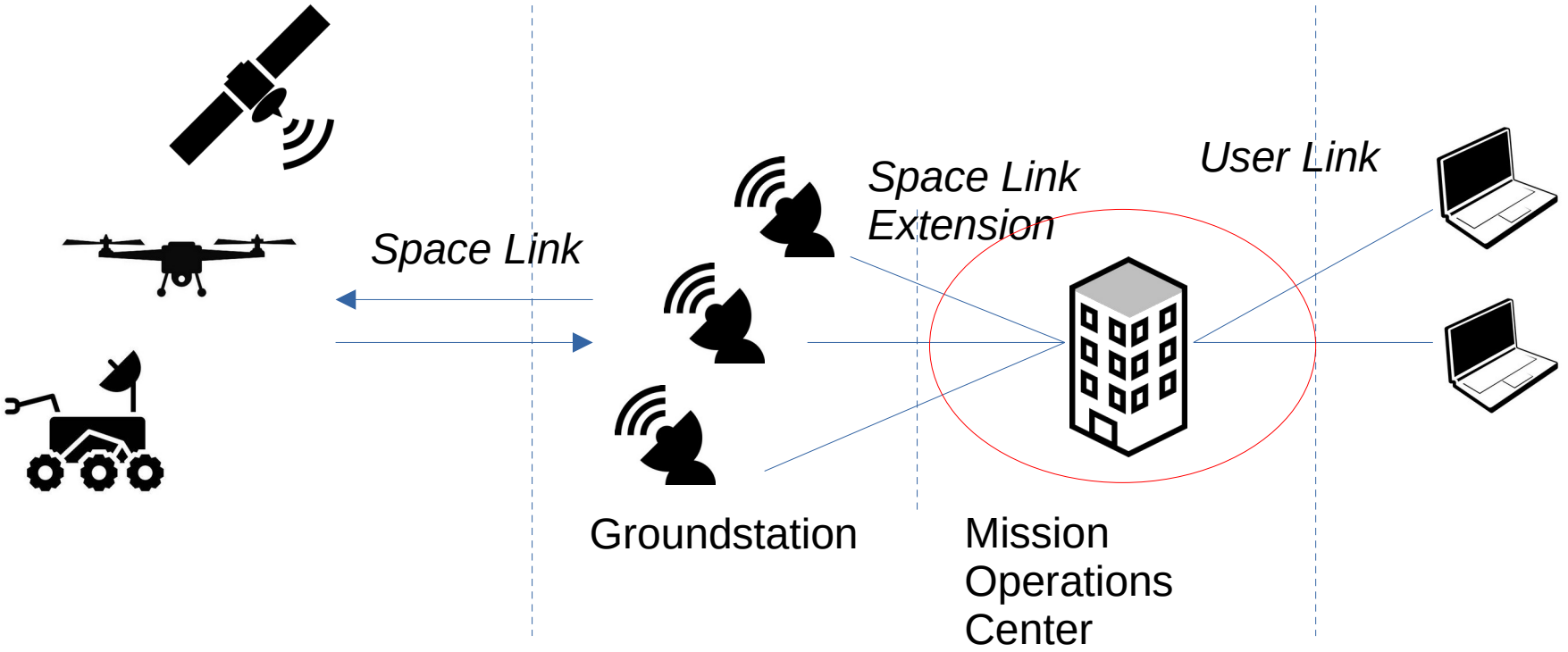
Telemetry	
Application Layer	- ECSS PUS Services - CCSDS File Delivery Protocol
Transport Layer	-/-
Network Layer	Space Packet Protocol
Data Link Layer (Protocol)	TM Space Data Link Protocol
Data Link Layer (Sync. And Channel Coding)	TM Sync. And Channel Coding
Physical Layer	RF and Modulation



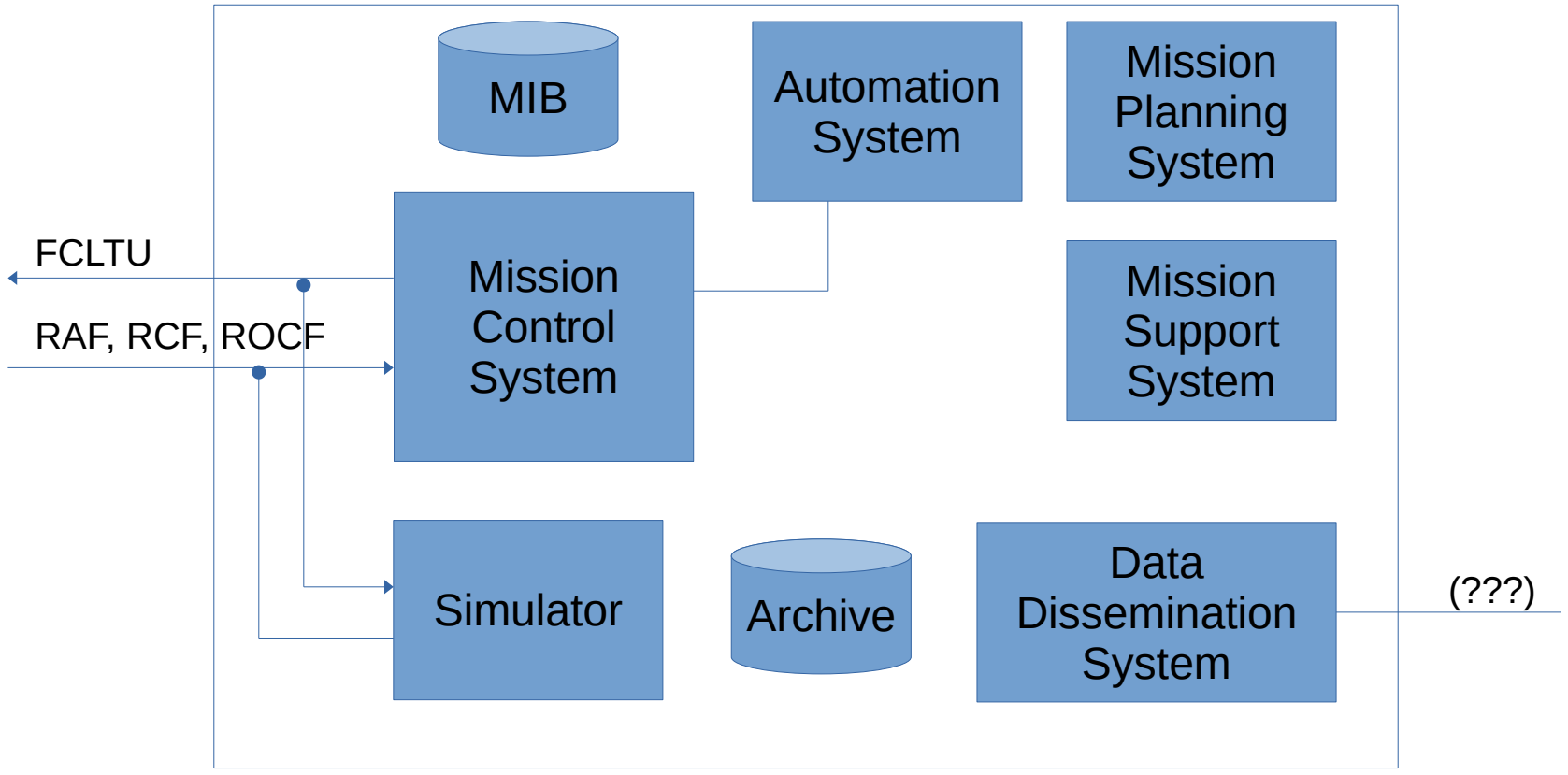
Segments


Space Link
Space Link
Extension
User Link
Groundstation
Mission
Operations
Center
Space/Remote Segment
Ground Segment
User Segment

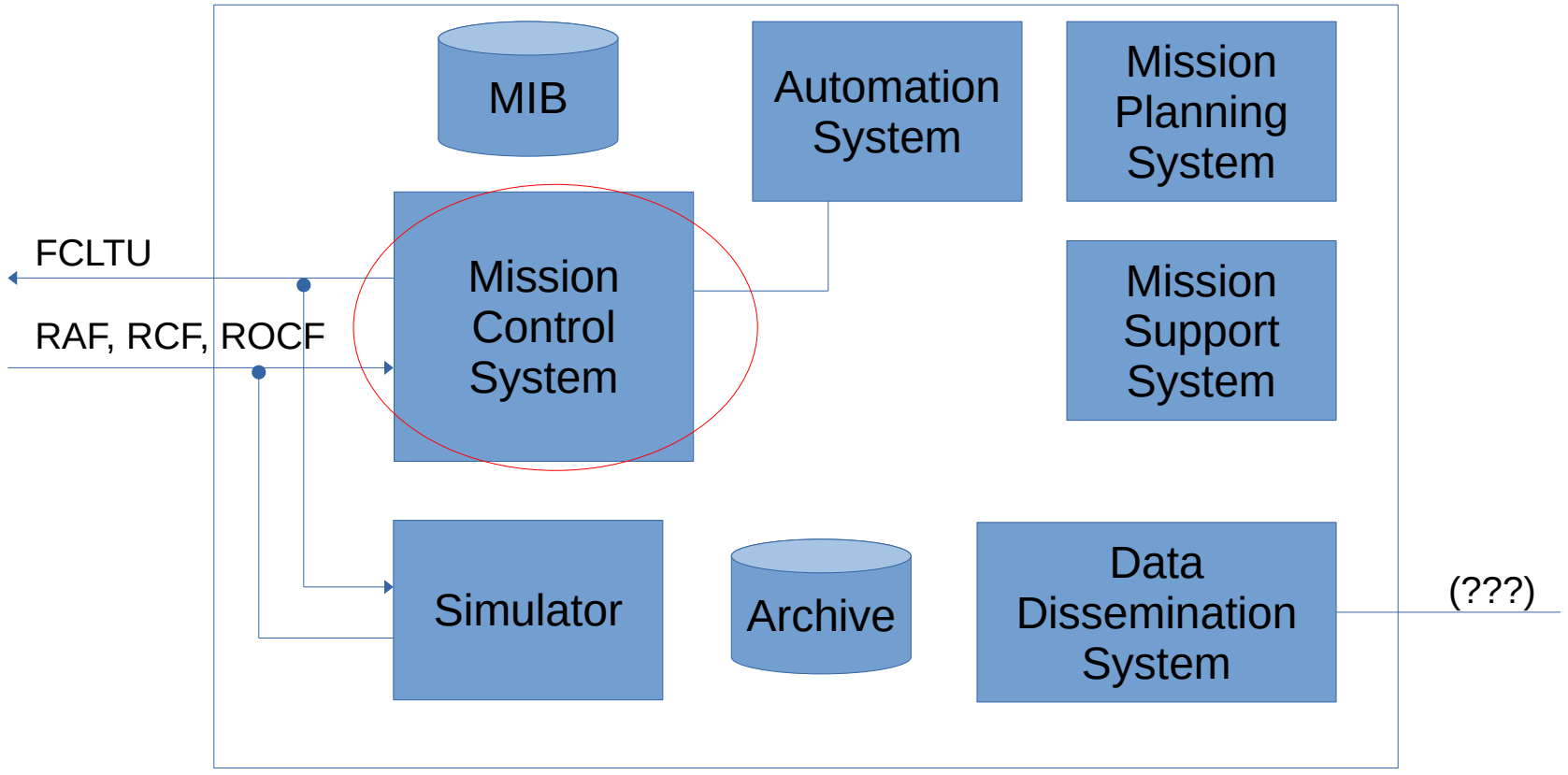
Segments


Space Link
Space Link Extension
User Link
Groundstation
Mission Operations Center
Space/Remote Segment
Ground Segment
User Segment

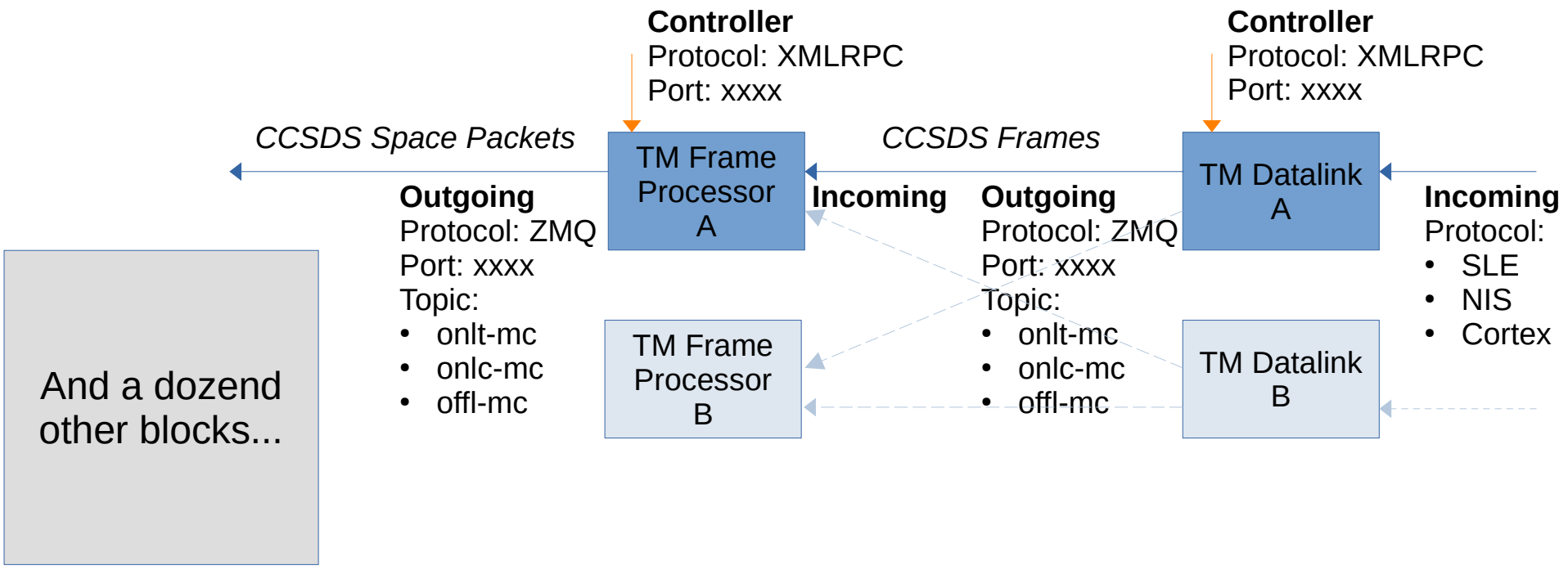
Mission Operations Center



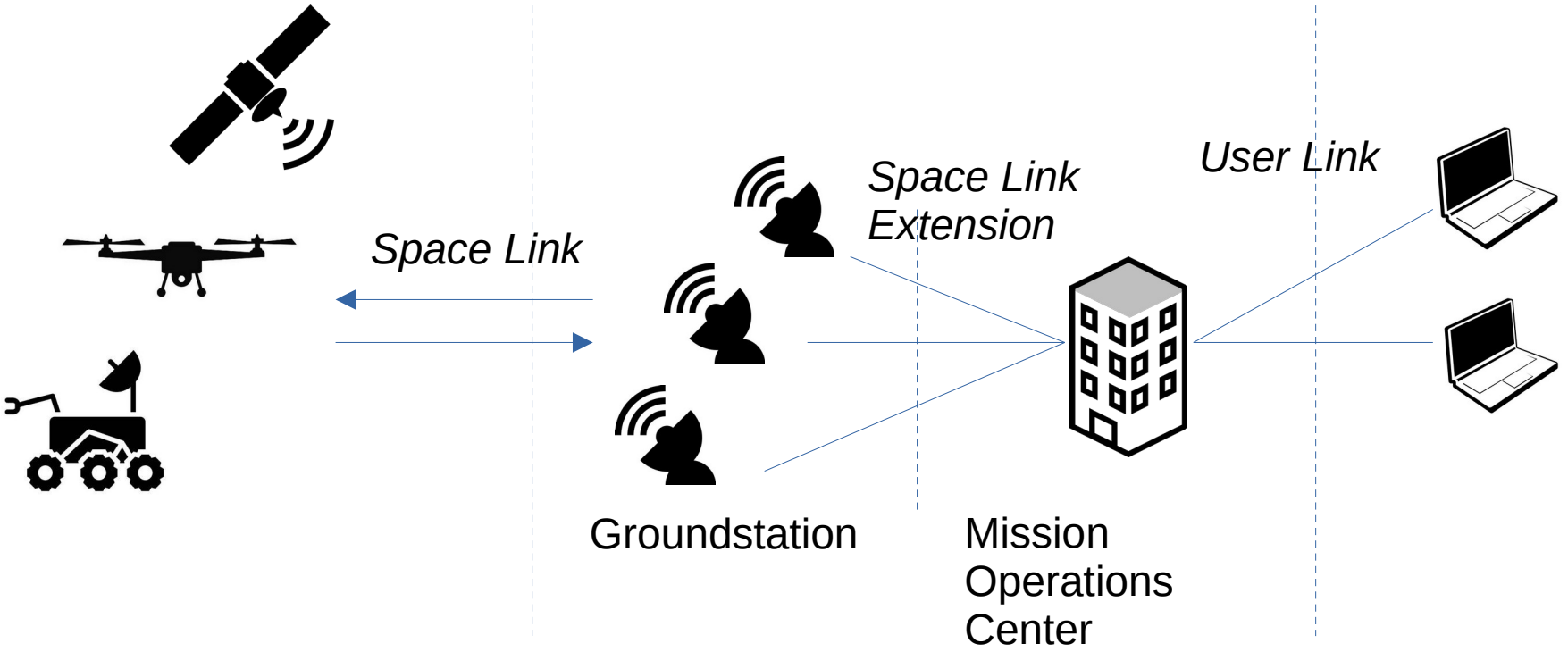
Mission Operations Center



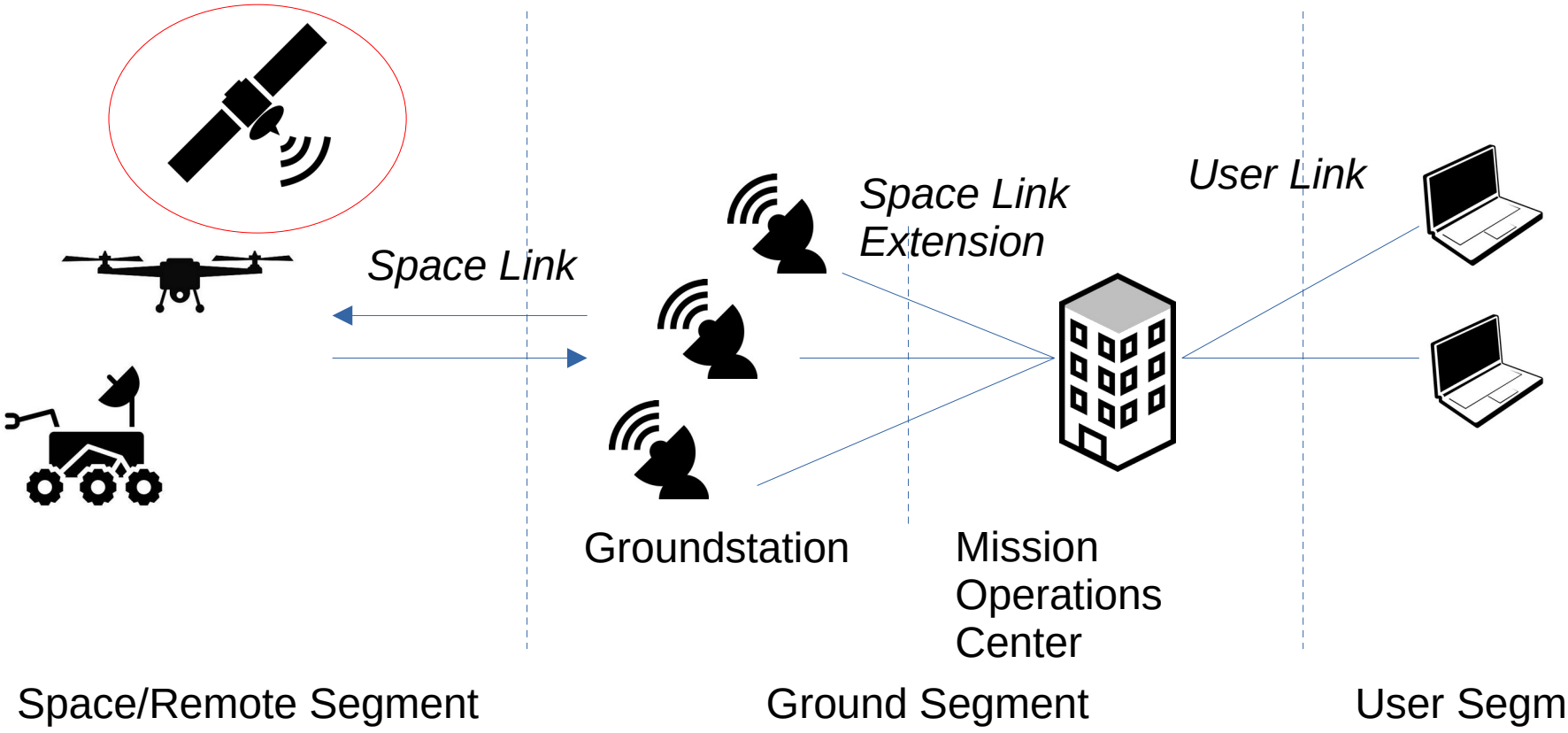
Mission Control System



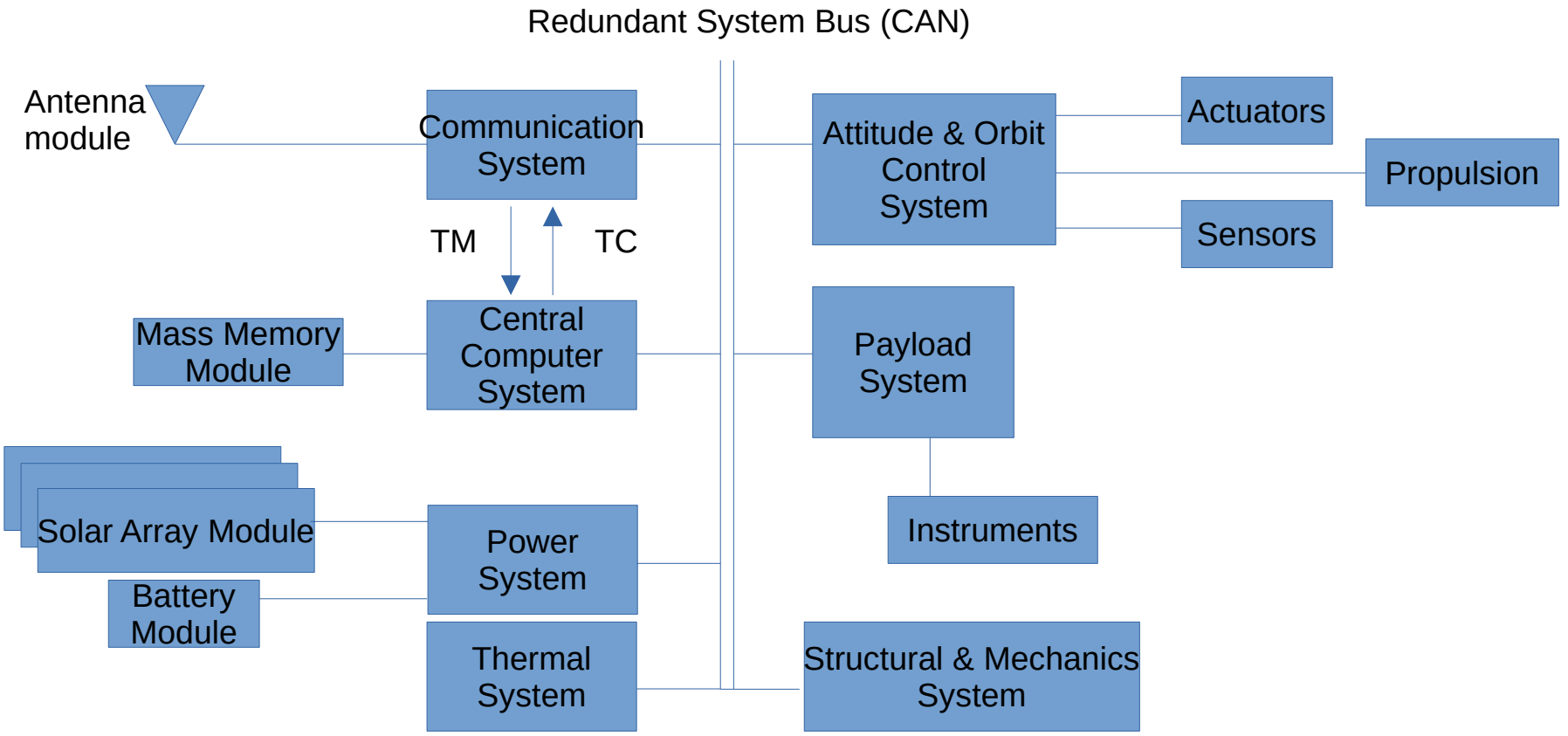
Segments


Space Link
Space Link Extension
User Link
Groundstation
Mission Operations Center
Space/Remote Segment
Ground Segment
User Segment

Segments

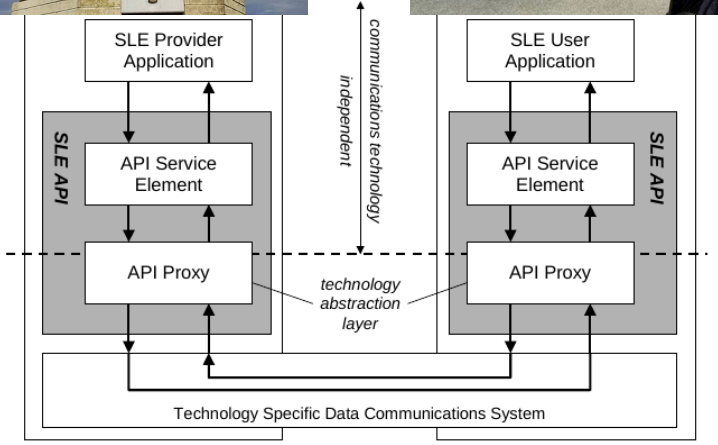

Space Link
Space Link
Extension
User Link
Groundstation
Mission
Operations
Center
Space/Remote Segment
Ground Segment
User Segment

CubeSat


Redundant System Bus (CAN)
Antenna module
Communication System
TM
TC
Mass Memory Module
Central Computer System
Solar Array Module
Battery Module
Power System
Thermal System
Attitude & Orbit Control System
Actuators
Propulsion
Sensors
Payload System
Instruments
Structural & Mechanics System

Projects of 2021 (some of them)

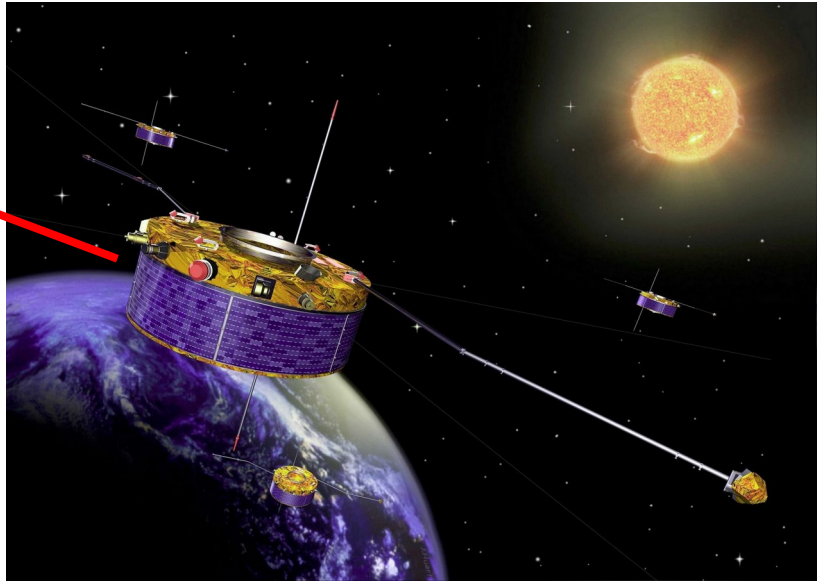
CCSDS SLE Protocol in Python




```python
import sle

raf = sle.RafUser(
    service_instance_identifier=os.environ.get('RAF_INST_ID'),
    responder_ip=os.environ.get('SLE_PROVIDER_HOSTNAME'),
    responder_port=int(os.environ.get('SLE_PROVIDER_TM_PORT')),
    auth_level='bind',
    local_identifier=os.environ.get('INITIATOR_ID'),
    peer_identifier=os.environ.get('RESPONDER_ID'),
    local_password=os.environ.get('PASSWORD'),
    peer_password=os.environ.get('PEER_PASSWORD')
)

raf.bind()
raf.start()
time.sleep(5)
raf.stop()
raf.unbind(reason='other')
```

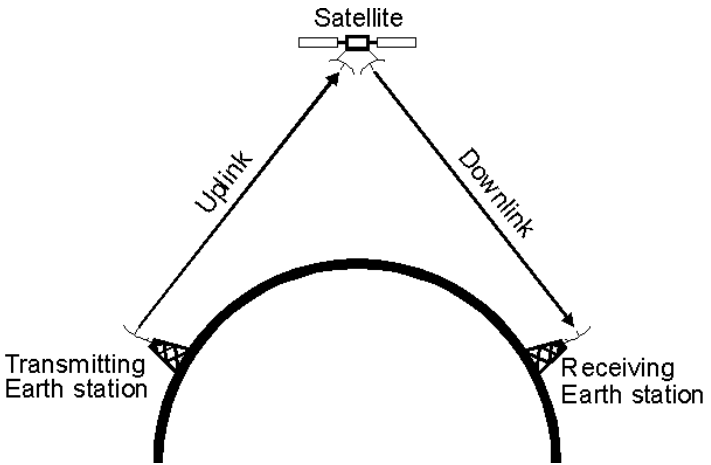
Example: NSPO tracks Cluster

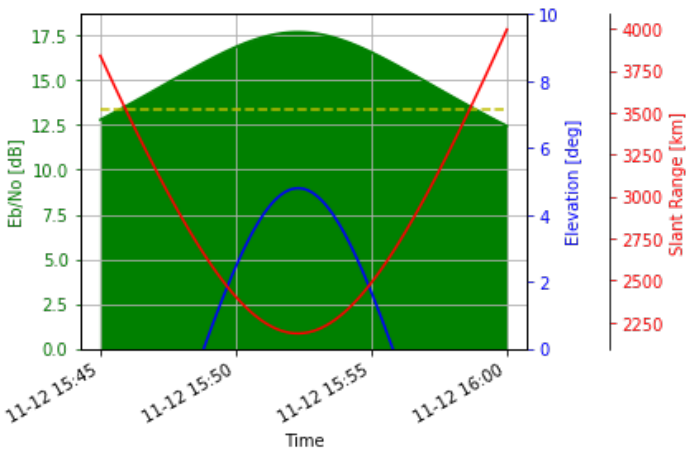
```
---------------Packet- 1 --------------------




Container:
    Header = Container:
        TransferFramePrimaryHeader = Container:
            FrameIdentification = Container:
                version = 0
                S/CID = 144
                VirtualChannelID = 2
                OperationalControlFlag = False
            MasterChannelFrameCount = 167
            VirtualChannelFrameCount = 254
            FrameDataFieldStatus = Container:
                SecHeaderFlag = True
                SyncFlag = True
                PacketOrderFlag = False
                SegmentationLengthID = 3
                FirstHeaderPointer = 2047
        TransferFrameSecondaryHeader = Container:
            SecondaryHeaderID = Container:
                SecondaryHeaderVersionNumber = 0
                SecondaryHeaderLength = 3
            SecondaryHeaderData = Container:
                AddVCFrameCounter = 70422
    TransferFrameData = Container:
        Data = b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\
xff\xff\xff\xff\xff\xff\xff'... (truncated, total 1096)
        OBT = b'&et\x86J\xca ' (total 7)
        RFB = 0
    CAB = 3


Data Length in Bytes 1096
```



Frames over SLE

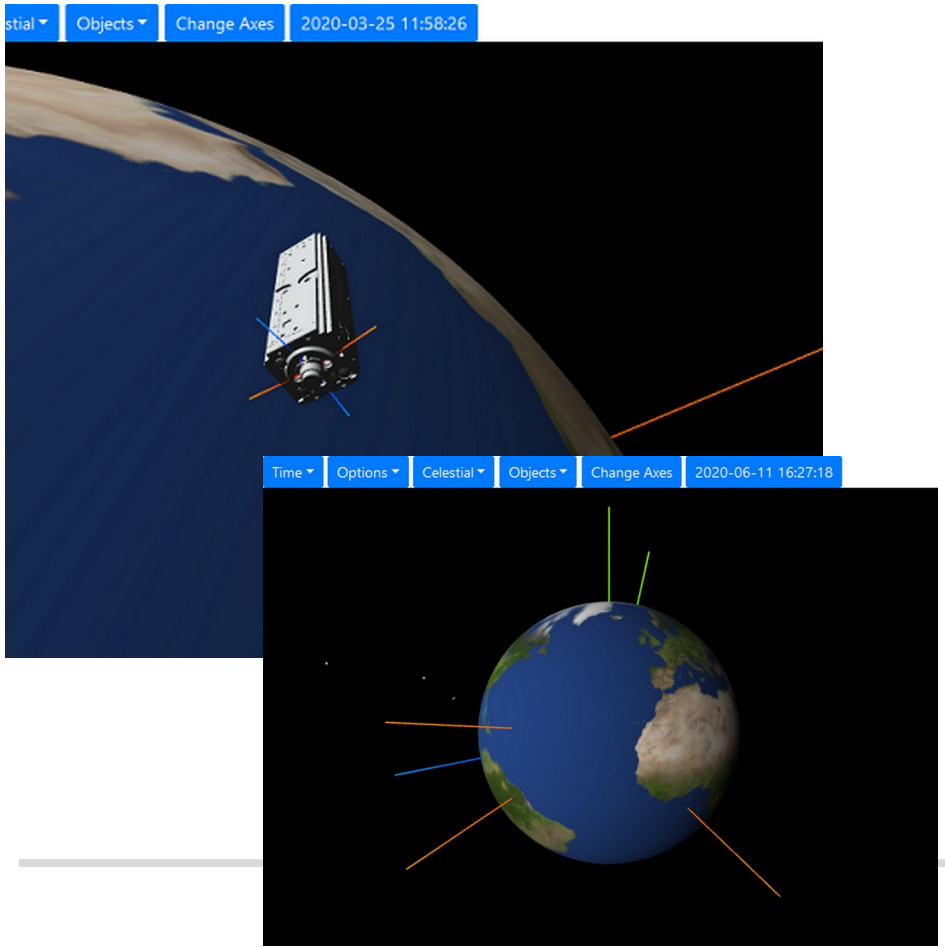
LinkPredict






```python
# Transmitter (on satellite)
onboard_losses = lp.Device(gain=-1)
amplifier_power = 0  # 0 dbW = 1 Watt
transmitter = lp.Transmitter(amplifier_power, [onboard_losses])
tx_antenna = lp.OmniDirectionalAntenna(gain=0, linear_polarized=True)

# Geometry
spacecraft = lp.SpacecraftObject()
name = "CUBEBEL-1 (BSUSAT-1)"
line1 = "1 43666U 18083E   18314.15998747  .00001095  00000-0  58587-4 0  9994"
line2 = "2 43666  97.5398 334.9753 0013890 232.5130 215.5169 15.17110642  1849"
spacecraft.set_orbit_from_tle([name, line1, line2])
groundstation = lp.GroundstationObject()
lat, lon, alt = 50.750, 6.216, 275
groundstation.set_location(lat, lon, alt)
geometry = lp.GroundstationSpacecraftGeometry(groundstation, spacecraft)

# Path
atmospheric_loss = lp.SimpleMediumLoss(1)
medium_losses = [atmospheric_loss]

# Channel
modulation = lp.FSKNonCoherentNoCoding(bit_rate=9600)
channel = lp.Channel(436.2e6,modulation=modulation)

# Receiver (groundstation)
line_losses = lp.Device(gain=-2.0)
receiver = lp.Receiver.from_noise_figure(noise_figure=2.0, devices=[line_losses])
rx_antenna = lp.MainLobeAntenna(peak_gain=15.5, beam_3db_width=5)
rx_antenna_noise = lp.SimpleAntennaNoise(300)
```

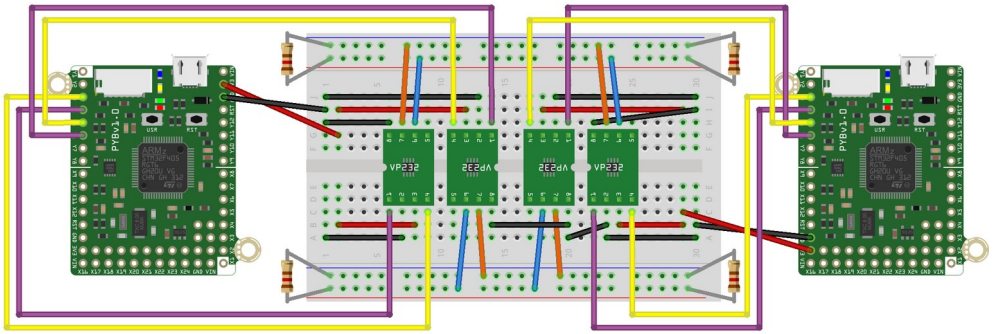
3D Visualization

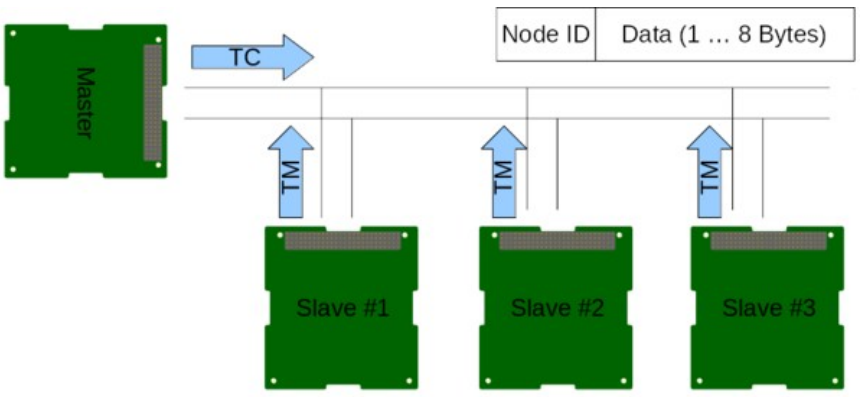



```
from browser import document, window, bind, timer
from datetime import datetime

from lib import three
from lib import camera
from lib import scene
from lib.celestial import EclipticGrid, Earth, Sun, Moon, Mercury, Venus,\
    Mars, Jupiter
from lib.satellite import Satellite


scene = scene.Scene()
scene.scene.add(three.AmbientLight(0x404040, 0.5))

camera = camera.Camera(scene)

# ecliptic grid
ecliptic_grid = EclipticGrid(scene)

# sun and planets
sun = Sun(scene)
earth = Earth(scene)
moon = Moon(scene)
mercury = Mercury(scene)
venus = Venus(scene)
mars = Mars(scene)
jupiter = Jupiter(scene)

camera.go_to(earth)
```

SpaceCAN

```python
def main():
    print("Master node started")
    pyb.LED(4).on()  # indicate the master node

    # switch bus via USER button
    def switch_callback():
        network.switch_bus()
        print("Bus switched")
        pyb.delay(200)  # short delay for switch debounce
    switch = pyb.Switch()
    switch.callback(switch_callback)
    print("Push button to switch bus")

    tick = time.ticks_ms()
    mode = 1
    while True:
        # check if telemetry from slaves arrived
        while telemetry.any():
            node, data = telemetry.read()
            print("Telemetry from {}: {}".format(node, data[:]))

        # send telecommand every few seconds to slaves
        if time.ticks_diff(time.ticks_ms(), tick) > 4100:
            tick = time.ticks_ms()
            print("Send telecommand, mode:", mode)
            for node_id in SLAVE_NODE_IDS:
                telecommand.send(node_id, [mode])
            mode = 0 if mode == 1 else 1
```



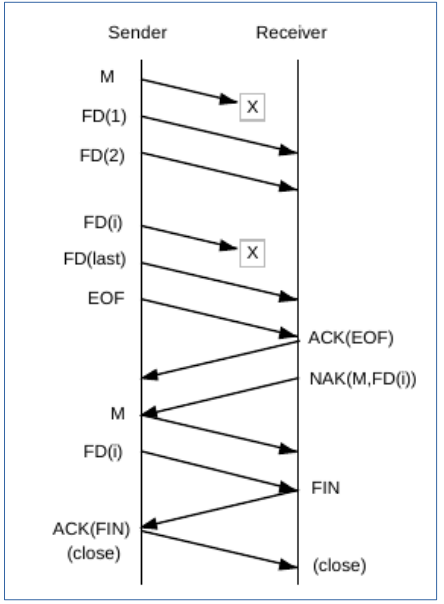


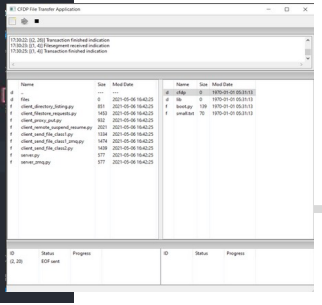
CCSDS File Delivery Protocol

LIBRE CUBE

Local Entity

```python
import logging
import time

import cfdp
from cfdp.transport import UdpTransport
from cfdp.filestore import NativeFileStore


logging.basicConfig(level=logging.DEBUG)

config = cfdp.Config(
    local_entity=cfdp.LocalEntity(
        2, "127.0.0.1:5552"),
    remote_entities=[cfdp.RemoteEntity(
        1, "127.0.0.1:5551")],
    filestore=NativeFileStore("./files/client"),
    transport=UdpTransport())

cfdp_entity = cfdp.CfdpEntity(config)

transaction_id = cfdp_entity.put(
    destination_id=1,
    source_filename="/medium.txt",
    destination_filename="/medium.txt",
    transmission_mode=cfdp.TransmissionMode.UNACKNOWLE

while not cfdp_entity.is_complete(transaction_id):
    time.sleep(0.1)

input("Press <Enter> to finish.\n")
cfdp_entity.shutdown()
```





Remote Entity

```python
import logging

import cfdp
from cfdp.transport import UdpTransport
from cfdp.filestore import NativeFileStore


logging.basicConfig(level=logging.DEBUG)


config = cfdp.Config(
    local_entity=cfdp.LocalEntity(
        1, "127.0.0.1:5551"),
    remote_entities=[cfdp.RemoteEntity(
        2, "127.0.0.1:5552")],
    filestore=NativeFileStore("./files/server"),
    transport=UdpTransport())

cfdp_entity = cfdp.CfdpEntity(config)
cfdp_entity.transport.bind()

input("Running. Press <Enter> to stop...\n")

cfdp_entity.transport.unbind()
cfdp_entity.shutdown()
```

Next up in 2022

- Starting weekly online meetings (Friday afternoon)
- Ground Segment
  - Mission Control System
  - Automation System
- Space/Remote Segment (best effort)
  - 1U Structure
  - 1U Power System

web: https://librecube.org
chat: https://matrix.to/#/#librecube.org:matrix.org
email: info@librecube.org

LIBRECUBE