SSC21-I-02

Autonomous Planning System (APS) for an Onboard TCPED Pipeline

Ken Center, Ella Herz, Evan Sneath, Sam Gagnard, Robert Glissmann, Rey Juarez, Neil Dhingra
Orbit Logic Incorporated
7852 Walker Dr, Suite 400, Greenbelt, MD 20770
ken.center@orbitlogic.com

ABSTRACT

As satellites and spacecraft grow in number and operate farther from Earth, there is an emerging need for increased autonomy via onboard decision making that is independent of ground stations but allows for collaboration between teams of assets. Such autonomy will relieve the burden on human operators, enable faster responses to dynamic events, and reduce communications between orbital assets and ground stations. Orbit Logic's Autonomous Planning System (APS) is flexible and customizable onboard software that enables teamed autonomy through the use of Tasking, Collection, Processing, Exploitation, and Dissemination (TCPED) pipelines onboard the satellites. Its small computational/memory footprint makes it especially suitable for small satellites: APS has been successfully demonstrated on constrained platforms such as the Raspberry Pi and the Unibap e2100. While APS is employed to create, plan and orchestrate TCPED pipelines, its flexible architecture allows it to interface with other satellite or software components that can provide states or events to inform or trigger planning, and to integrate with satellite resources that can execute those plans. For example, in an Earth-imaging satellite mission, APS tasks the satellite to perform collections, facilitates delivery of the collected data to onboard processing/analysis modules, and uses the results to inform future tasking, e.g., following-up with additional collection or processing. APS on a given asset employs one or more Specialized Autonomous Planning Agents (SAPAs), software modules that plan onboard activities for a specialized need. Through configurable plugins, they can be customized to the capabilities and mission roles of the host asset. Each SAPA is dedicated to a general mission- or system-level need (e.g., separate SAPAs may focus on collection planning, contact scheduling, and fault management) and issue one or more high-level activities to fulfill that need. These activities are fielded by the Master Autonomous Planning Agent (MAPA), which performs intelligent deconfliction of the onboard resources that activity execution requires. The resource execution timeline is composed to maximize the "goodness" of all competing activities using a configurable multi-factor figure of merit (FOM). APS's modular architecture and well-defined interfaces facilitate rapid development and deployment of novel or enhanced capabilities. The level of autonomy is customizable and can be tuned over the course of the mission to allow the satellite more autonomy as it gains trust. These features allow APS to be easily deployed for complex satellite missions with multiple competing mission objectives. APS's constellation-level collaborative autonomy seamlessly extends its asset-level autonomy. Multiple APS-enabled satellites equipped with inter-satellite links or access to a space network can coordinate without ground station communications, e.g., a constellation of imaging satellites can perform load balancing among themselves to ensure coverage and limit redundancy. Such autonomous collaboration is especially important in scenarios where evolving conditions change mission parameters, e.g., if one satellite collects imagery from a region, and processing of that imagery identifies signatures warranting follow-up tasking, a different satellite overflying the location in the near future can perform the collection. APS has been developed and extended for multi-domain, multi-asset mission applications through multiple programs sponsored by AFRL, DARPA, NASA, and ONR.

INTRODUCTION

Planning and scheduling for traditional spacecraft operations occur on the ground and the resulting instructions are uplinked to the spacecraft for execution. This procedure has inherent time delays and such a centralized commanding architecture imposes drawbacks whose impacts compound with scale, e.g., as spacecraft operate further from Earth and as they operate in larger constellations. Communication latency limits spacecraft operations, potentially causing the loss of critical mission opportunities or even of the spacecraft itself. These issues compound as spacecraft responsibilities comprise more of a Tasking, Collection, Processing, Exploitation, and Dissemination (TCPED) pipeline where tasks are coupled through dependent links. Although the completion of individual tasks may not take a long time, significant latency would be accrued in waiting for coordination from a ground station between each pair of steps in the TCPED pipeline. Moreover, constellation operations are sensitive to any issues that affect ground control, from communications errors to processing overload to weather conditions and beyond.

The availability of more capable flight hardware and inter-satellite links (ISLs) provides a way to address these shortcomings through onboard planning and scheduling and distributed commanding architectures. The distributed commanding architectures enabled by such onboard software allows spacecraft to detect opportunities on-orbit and react immediately without needing to communicate to the ground station and wait for the response plan. Faster response times lead to more effective response actions and enhanced mission success. Using ISLs, the right planning software can orchestrate communication and coordination among constellation elements to enable *collaborative* autonomy. This enables spacecraft to cue one another for collection or other steps of the TCPED pipeline, and for spacecraft to autonomously optimize activity schedules at a constellation level.

Orbit Logic has developed the Autonomous Planning System (APS), software that can run onboard spacecraft and enable them to respond to onboard and external events to meet the planning/scheduling requirements of a variety of missions. Its modular architecture allows planning systems to be assembled from individual planning components and quickly configured (and reconfigured as necessary) to meet initial and dynamic mission goals. APS operates using a rolling timeline, constantly adding or modifying the existing spacecraft Command Queue as new information is received in the form of dynamic and frequently ad-hoc events.

APS has a small footprint that allows it to be deployed onboard small satellites or other robotic agents with modest computing power. APS has uses beyond satellite operations; it can enable autonomy on any platform and can enable collaborative autonomy on any group of robotic agents that can communicate amongst themselves. The APS architecture reduces the cost, schedule, and risk of implementing planning systems – making asset more able to respond to dynamic mission goals and more efficient with the use of their processing resources. Beyond various satellite programs, Orbit Logic has applied APS to teams of robotic vehicles in underwater and space exploration missions.

The remainder of the paper is organized as follows. First, we provide background on traditional, centralized satellite mission scheduling, autonomous spacecraft operations, and distributed commanding architectures. We then outline a motivating scenario that will provide a concrete touchstone for the rest of the paper. Using this motivation, we then describe the architecture of APS as installed on each asset; broadly speaking. The following section provides details on the APS architecture, including how constellation different assets interact for distributed planning, how APS is deployed on assets, and how the APS Mission Executive module facilitates in-situ management of the APS on-asset configuration once deployed. After these technical details, we discuss APS for enabling an autonomous on-board TCPED pipeline. We then return to the motivating scenario and present simulation results illustrating the efficacy of APS. Moving beyond this scenario, we then discuss several different deployments of APS and illustrate the breadth of its potential. Finally, we provide concluding remarks.

BACKGROUND ON COMMANDING ARCHITECTURES

Background on Satellite and Spacecraft Mission Planning and Scheduling

In traditional operations, satellite mission planning is performed at ground stations and the resulting schedules are transmitted to satellites via command load during periods of communications availability, generally during periodic ground station contacts. This approach inserts significant delays in responding to new information and new opportunities. These delays can range from several minutes to several hours and can cause the loss of opportunities for the collection of critical intelligence data, the degradation of the satellite due to a slow response to a system failure, or even the loss of the satellite (as in the case of an orbital collision between two satellites). These communications latencies can arise due to adherence to pre-planned contact schedules or speed of light delays.

Latency due to contact scheduling is an artifact of spacecraft commanding architectures and it is possible, but certainly nontrivial, to limit it. Figure 1 shows a traditional mission planning timeline, where planning is constrained by contact opportunity timelines. In such an architecture, satellite command load can only address specific events if the mission planning group knows about the event prior to the start of the planning process. This concept of operations is not equipped to respond to opportunities that are detected between commanding opportunities, which can be several minutes or even hours apart.

Latency due to the speed-of-light is unavoidable and can only be addressed through autonomous decision making; for example, future spacecraft performing geyser-monitoring missions on Europa and/or Enceladus would

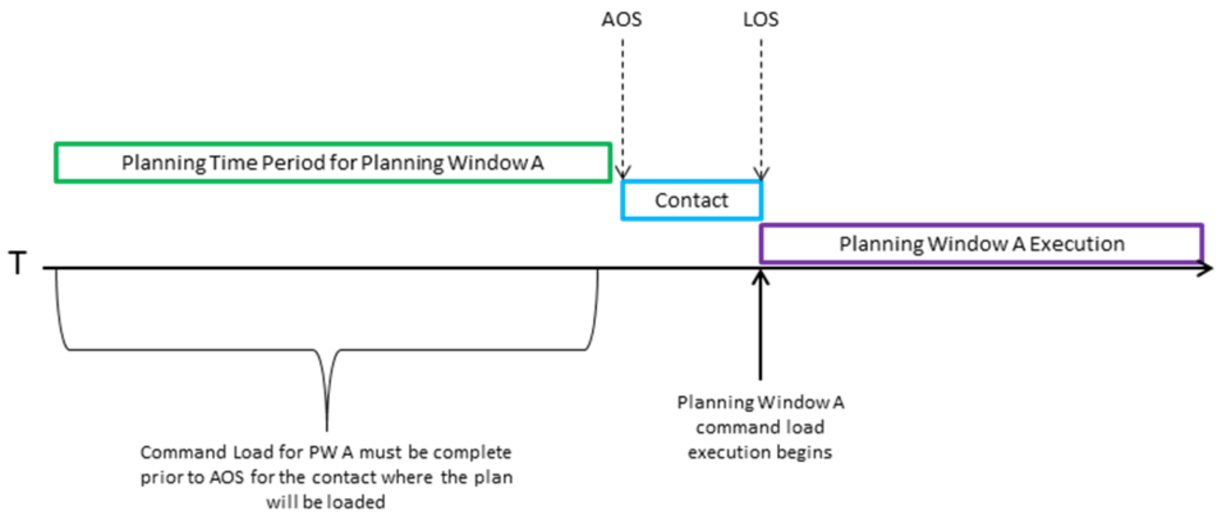
AOS
LOS
Planning Time Period for Planning Window A
Contact
Planning Window A Execution
T
Command Load for PW A must be complete prior to AOS for the contact where the plan will be loaded
Planning Window A command load execution begins

Figure 1. Traditional Ground-Based Mission Planning Timeline with contact occurring between Acquisition of Signal (AOS) and Loss of Signal (LOS).

need to operate autonomously to react to geyser events (e.g., detect, navigate, collect samples, etc.) lest communications latency to Earth result in missing the data gathering opportunity. A planning agent, such as APS, located onboard the spacecraft eliminates this major deficiency of the existing mission planning concept of operations by making appropriate changes to the existing spacecraft command load (or generating a new load from scratch) in response to events in near-real-time. The onboard planning agent can generate and/or maintain a plan that responds to evolving conditions between ground uplink opportunities. Migration of mission planning activities to an autonomous flight software agent will allow future missions to implement true real-time opportunistic target collections, and other unrealized capabilities enabled by onboard planning.

Recent Advances in Spacecraft Autonomy

Some satellite missions have implemented onboard autonomy, such as Air Force's TacSat-3[1] and NASA's EO-1[2] missions; however, the solutions implemented in those and similar missions are typically mission-specific so could not be considered a modular architecture easily adaptable to new missions, and/or use state-based and rules-based planning that cannot scale to meet complex planning needs within the constraints of onboard resources. Recent work at NASA[3,4] addresses the growing need for autonomy in fault detection; however, there are still significant gaps. In Frank et al.[3], station operations were moved onboard – from the ground to the ISS – but with lots of crew-in-the-loop operation. The system tested in Aaseng et al.[4] performs complex power planning in the presence of power system faults, but these faults are previously characterized and there is no infrastructure for learning to identify and respond to new classes of events.

The APS Distributed Architecture

The APS architecture[6, 11] consists of Specialized Autonomous Planning Agents (SAPAs) that address specific planning needs (recorder management, ground target imaging, collision avoidance, etc.) and a Master Autonomous Planning Agent (MAPA) that ingests the output of the SAPAs, deconflicts global resources, and creates a final plan that it forwards to the onboard task executive for implementation. This unique approach contrasts with the current state-of-the-art for planning systems which generally try to apply a single algorithm type (often state-based and rules-based for flexibility) to multiple planning domains in a one-size-fits-all approach, which often results in suboptimal planning.

The autonomous planning ability enabled by APS allows satellites to respond much more quickly to capture opportunities that might otherwise be missed. The MAPA/SAPA onboard architecture offers the flexibility to plan for different kinds of opportunities, keeps the system modular and efficient enough to be used in constrained computing environments, and makes the system extensible to almost any satellite planning domain.

The MAPA/SAPA architecture and planning timelines lend themselves to coordinated constellation planning because the individual components do not care where the event messages originate (on the same satellite, a different satellite, or the ground), and planning can be performed and re-performed as different systems react to the environment as understood from event messages on the bus. As more spacecraft need to coordinate activities

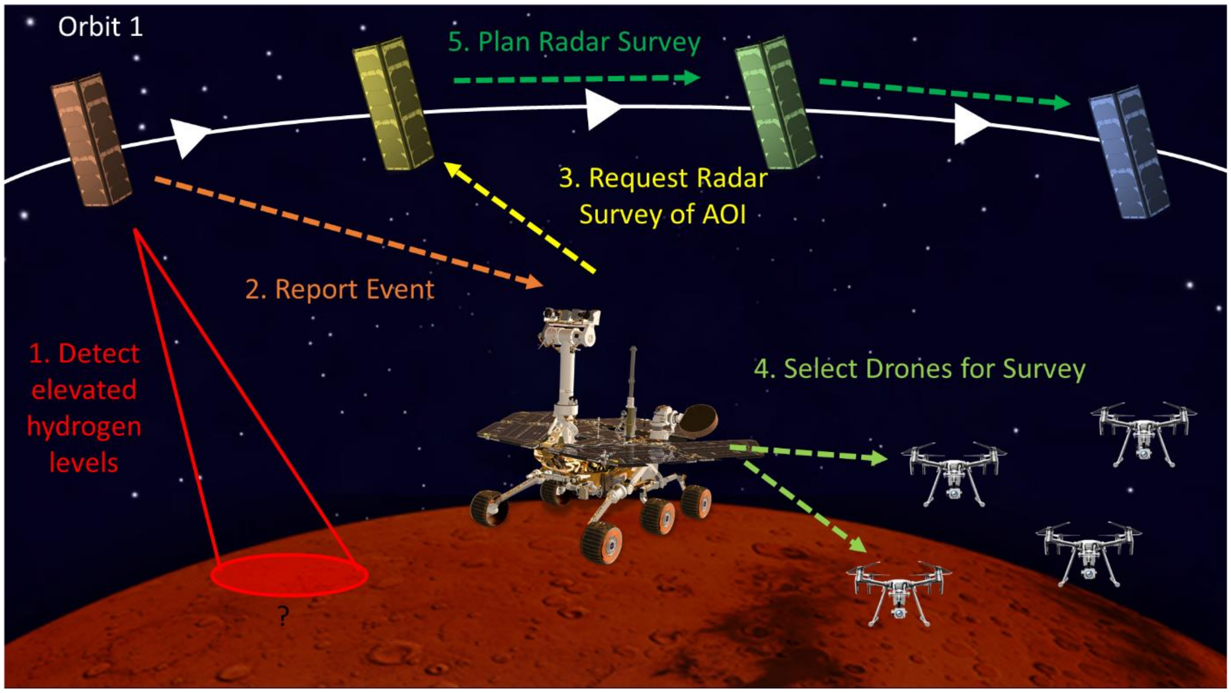
Orbit 1
5. Plan Radar Survey
3. Request Radar Survey of AOI
2. Report Event
1. Detect elevated hydrogen levels
4. Select Drones for Survey
?

Figure 2. Sample Use Case Steps 1-5.

to reach specific goals as a whole, a configurable and adaptable planning architecture becomes more critical.

MOTIVATING SCENARIO: COLLABORATIVE ROBOTIC EXPLORATION OF MARS

To provide a concrete motivational example, we will present APS in the context of our Mars/Interplanetary Swarm Design and Evaluation Framework (MISDEF) program. This NASA program was inspired by several swarm scenarios of interest from the 70th International Astronautical Congress proceedings[11]. One sample use case, shown in Figure 2 and Figure 3, involves a swarm of orbiting assets, drones, and a rover.

Sample Use Case Description:

1. One Satellite in the orbiting satellite constellation identifies an Area of Interest (AOI) based on its detection of elevated hydrogen readings during a coordinated survey of the Martian surface.
2. The Satellite communicates the detection event to the coordinating Rover at the next available opportunity, potentially via relay through other constellation members.
3. The Rover queries stored radar and map data and determines that there is no radar data for the AOI. However, the region is determined "accessible" by previous map data. The science is determined valuable enough that the Rover will request a radar scan of the region during the following orbit. The Rover conveys the location of the AOI and the desired scan request to the next available satellite.
4. Meanwhile the Rover requests that several Drones return to the Rover, as a result of a high science area being deemed "accessible". Upon return to the Rover, Drones correct their location and transfer data to the Rover to clear their storage.
5. The Satellite that received the AOI location and scan request determines which orbital assets will perform the radar scan, as well as other scan specifics (when the scan will be performed, attitude maneuvers, scan parameters).
6. The instructed Satellite performs its radar scan of the AOI.
7. The Satellite constellation returns the radar data to the Rover. The radar data can be analyzed to determine the surface hardness of the crater, indicating which areas of the crater should be chosen for sample collection.
8. The Lander distributes the survey plan to the Drones, including which Drones will complete which survey tasks (sample collect,

environmental survey, Imaging).

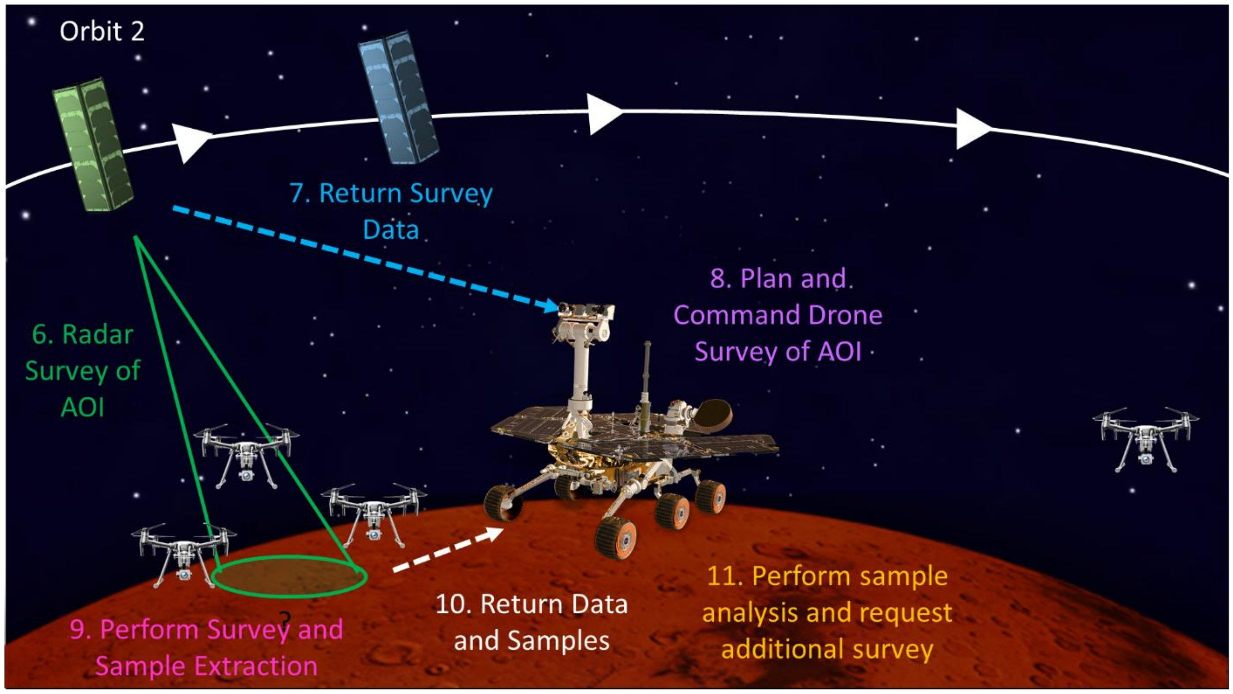
Orbit 2
7. Return Survey Data
8. Plan and Command Drone Survey of AOI
6. Radar Survey of AOI
9. Perform Survey and Sample Extraction
10. Return Data and Samples
11. Perform sample analysis and request additional survey

Figure 3. Sample Use Case Steps 5-11.

9. The Drones execute the survey plan.
10. The Drones return to the Rover and deposit their samples and data.
11. The Rover performs analysis on the samples and data, determines that the data reports samples with higher likelihood of water and requests an additional survey into the area.
12. The Rover will summarize these and other science activities and periodically send results through the orbital constellation back to Earth.

AUTONOMOUS PLANNING SYSTEM (APS) ARCHITECTURE

APS is a decentralized software architecture that can be distributed across multiple assets (e.g., satellites) in a heterogeneous swarm. Each APS instance has access to configurable plugins modelling resource capabilities, vehicle mission activities, and Specialized Autonomous Planning Agents (SAPAs) – software modules that orchestrate onboard activities for a specialized need – that are particular to the host asset and its capabilities. Each SAPA is dedicated to a general mission or system-level need (e.g., to monitor a region of the Earth) and issue one or more high-level activities needed to fulfil that need. These issued activities are fielded by the Master Autonomous Planning Agent (MAPA), which focuses on intelligent deconfliction of the available resources that are capable of carrying out the given activity.

Specialized Autonomous Planning Agents

Each instance of APS is equipped with platform- and mission-specific SAPAs to perform particular functions using tailored algorithms. For example, an imaging collection SAPA would understand the parameters of the sensor, imaging modes, types of targets, considerations such as weather and incidence angle, etc. The SAPA elements of the APS architecture give it flexibility since it allows different types of planning routines, algorithms, and considerations for different kinds of operations. Each SAPA can perform planning for a different kind of operation (imaging vs. orbit maneuvering vs. downlink planning), and the MAPA is used to generate the integrated, deconflicted multi-SAPA schedule for execution. To illustrate the strength of this approach, we discuss four different SAPAs here

- Ground Observation SAPA
- Communication SAPA
- Processing SAPA
- Fault Learning Agent for Prediction, Protection and Early Response (FLAPPER).

The basic process flow that all SAPAs follow is shown in Figure 4. The execution flow involves work item

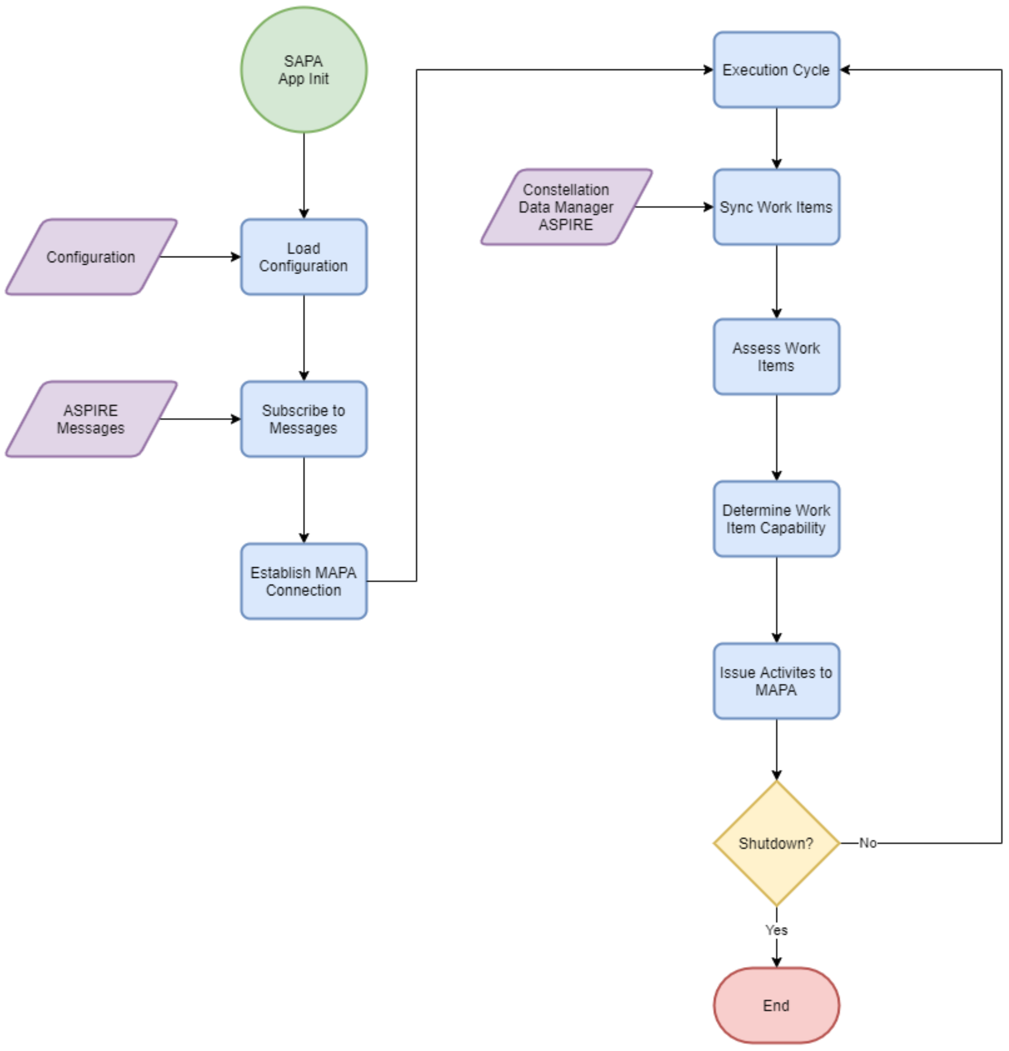

Figure 4. SAPA Workflow.

assessment translated to activity needs and activity capability assessment. Spawned activities are then assessed in the MAPA as they are received for resource-level deconfliction.

We now discuss several example SAPAs. These do not constitute an exhaustive list of the SAPAs that Orbit Logic has developed or the capabilities that APS can support, but they form a representative sample of SAPAs that are relevant for space operations.

Ground Observation SAPA

This SAPA plans ground image collections using multiple supported sensor types as they are available on the satellite platforms. This may include sensor types such as Synthetic Aperture Radar (SAR), Electro-Optical (EO), or Infra-Red (IR) imagery. The collection schedule is driven by the latest satellite ephemeris data available onboard, the latest set of targets of interest (provided by event messages from onboard or external sources like the ground or other satellites), and available sensors and their attributes.

The Ground Observation SAPA has specialized internal logic and algorithms to compute access times from configurable sensors to ground targets of interest during the current rolling planning period. Using a figure-of-merit, the most valuable constraint-conforming collection time and mode will be selected within configurable imaging buffers. For situations where multiple targets have overlapping or conflicting beam access, the same figure-of-merit is used to select the most valuable target acquisition with the lesser merit collection being shifted to a deconflicted window. At the end of the process, the Ground Observation SAPA outputs a series of high-fidelity target acquisition activity requests for the current planning period to be consumed and deconflicted by the MAPA to resolve spacecraft resource-level constraints. The SAPA retains ground observation target fulfillment status for all identified targets for use during future planning windows, and updates that fulfillment status based on messages from the MAPA or telemetry from other external flight software components. Updated target status is reported to the APS Constellation Data Manager component for syncing of work item state to all constellation and ground entities.

Communication SAPA

This SAPA plans the exchange of data via communication links between constellation and ground assets. This is not only required for periodic connection with ground station, but also used for dissemination of information between constellation assets and transfer of data resulting from constellation work items for further processing or downstream tasks on other constellation assets. For instance, a stereoscopic collection between two spacecraft may require onboard processing and analysis of the stereoscopic image pair. In this case, at least one of the stereoscopic images must be transferred to a constellation asset capable of processing stereoscopic imagery. The Communication SAPA is used in this case to model, plan, and facilitate this transfer of information.

The Communication SAPA models constellation asset positions based on propagated states as well as the communication sensor capabilities of each of these assets. This information along with geometrical data informs the vehicle of possible access windows between assets. If it is determined that data must be transferred from one constellation asset to another, the Communication SAPA will then plan the best period in which the transfer of data can be carried out. The newly-spawned communication activity requested from the SAPA will be assessed for a figure of merit, and issued to the MAPA for further vehicle-level resource deconfliction. When successfully deconflicted, this activity then follows the standard path of being added to the internal schedule of resource constraints and issued out as a fully realized plan by APS for execution by an onboard timeliner component.

Processing SAPA

The objective of this SAPA is to perform onboard processing on a constellation asset for any given onboard processing algorithm while modeling processing capabilities and resource loads and requirements for the vehicle. Based on these configurable conditions, the vehicle is then able to both bid on processing-related work items in the list of shared constellation work items based on onboard processing capabilities and work item needs.

Similar to other SAPAs mentioned, relevant work items are assessed against asset capabilities and, if capability is found, relevant processing activities are planned and requested to the MAPA based on estimated availability of processing algorithm input data. This processing activity is rated based on multiple figures of merit (most notably on the time required to perform the processing given available memory and computing resources) and then further deconflicted at a vehicle level in the MAPA. The Processing SAPA is often paired with a data collection SAPA such as the Ground Observation SAPA and optionally may be paired with the Communication SAPA for heterogeneous constellations where vehicles collecting data may not have capability to process the collected data.

Fault Learning Agent for Prediction, Protection and Early Response (FLAPPER) SAPA

This SAPA is currently under development and will enable autonomous fault management that leverages Machine Learning (ML) capable of detecting, isolating, and mitigating anomalies in real- or near-real-time with minimal ground intervention. A set of defined fault detection and correction constraints will be employed, along with the capability for operators to classify new types of faults and responses. These constraints, along with spacecraft data input, will be used to train the FLAPPER Fault Detection Service to detect and classify faults and their corresponding responses based on novel telemetry limits and value trends. The scheduling component of the FLAPPER SAPA will subsequently plan correlated corrective actions. An initial prototype of FLAPPER was demonstrated on flight-like hardware and tested on telemetry from the NASA Lunar Atmosphere and Dust Environment Explorer (LADEE) mission. The FLAPPER ML model was run on the Unibap e2100 flight processor that Orbit Logic utilized on AFRL’s Resilient Bus Experimental Laboratory (REBEL) Testbed, which is representative of hardware supporting many current mission concepts.

The FLAPPER ML engine was also validated in an experiment replicating the NMS (Neutral Mass Spectrometer) temperature failure that occurred on LADEE February 26, 2014. The process that was performed to train the model is similar to what would be performed during mission development and in operations. First, a TensorFlow model was configured with ML parameters and configurations. Then telemetry features that represented the NMS temperature state were identified. Training for this model took approximately 10 mins for the 6 features for a month-long period of healthy spacecraft telemetry. Subsequent to the training an anomaly threshold was chosen. Injecting the mission data in playback fashion from a four-month timeframe surrounding the NMS temperature event, the ML Trained Model detected anomalous behavior in mid-November 2013 (green line), 3 months prior to the detection of the event (orange line) during the actual mission operations (see Figure 6).

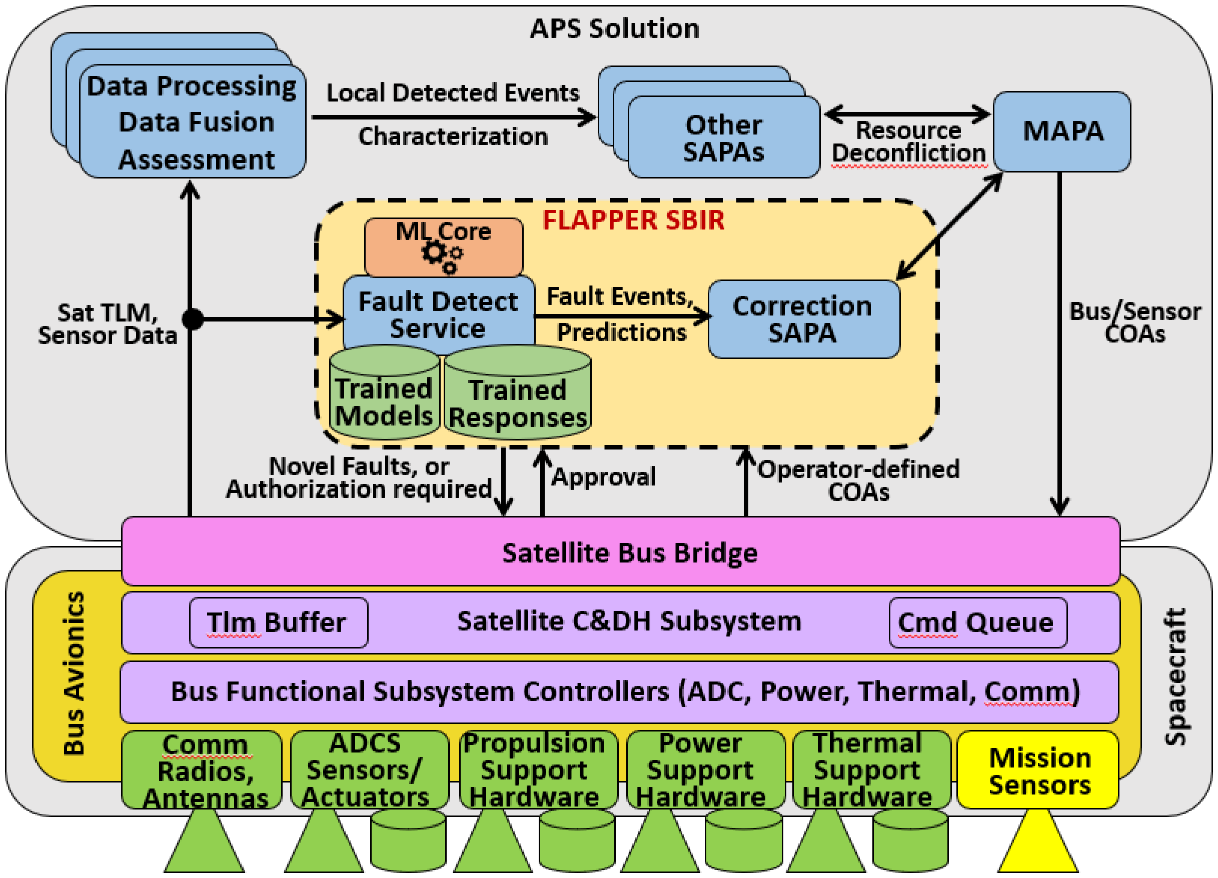

Figure 5. APS architecture with FLAPPER for machine-learning-driven fault detection and correction

The Master Autonomous Planning Agent

After capability windows are determined by SAPAs - given the high-level activities, derived actions, and any timing or geometric constraints, the individual activity actions are levied against the resources required to perform them. Each resource assesses its ability to carry out the action alongside the previously committed actions in the schedule – which results in resource deconfliction through action shuffling or merging. The "goodness" of an activity to be inserted into the timeline is scored using a configurable multi-factor figure of merit (FOM). Activities are scheduled in a rolling timeline, and fully resolved and scheduled activity actions are submitted over a messaging interface interpretable by an onboard command timeliner service. Resources in APS maintain an internal timeline of all committed activity actions and through this are able to provide lookahead appraisal of resource usage at a desired time that is grounded in the various subsystem states as reported in spacecraft telemetry.

Mission flexibility is a central tenant of APS. This includes the flexibility to enable advanced concepts of operation by supporting multiple behaviors within the lifetime of a mission, support for development of new mission needs per operational program, support for flexibility in sensor and onboard resource types across any operational domains, and flexibility to execute APS on various computing architectures with the intention to minimize overhead in both computation cycles and memory footprint. These APS capabilities are achieved through a robust set of core services, tools, and base functionality which are used across missions. From these tools, SAPAs and resource models specific to mission needs can be developed and applied as-needed to a heterogenous set of constellation assets based on vehicle capabilities. For example, a space vehicle with an EO/IR imager may be defined with a SAPA specializing in ground location observation, while a ground vehicle may have APS executing with a SAPA specializing in terrain mapping via LIDAR sensing resources. Both missions in this case would be executing the same core APS application, but would utilize a different set of SAPAs and resources. The advanced planning, scheduling, and deconfliction capabilities are identical. This makes APS ideal for complex mission examples with connected constellation assets across domains where each asset needs to be configured to meet specific roles at different points in the mission timeline. Examples of resource

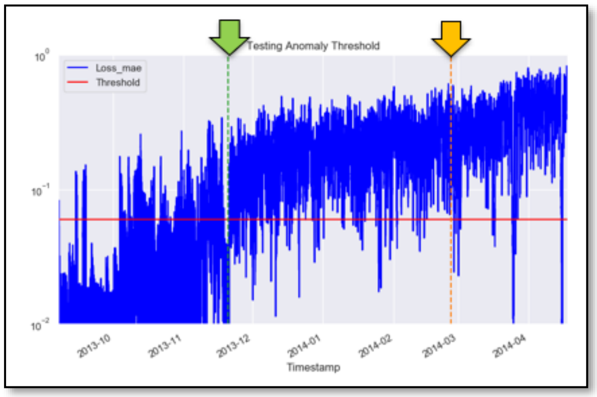

Figure 6. Loss MAE with NMS fault telemetry from LADEE Mission; orange arrow indicates when the fault occurred and green arrow indicates when FLAPPER detected it

configuration point include sensors, actuators, data storage, communication, and processors.

APS IMPLEMENTATION

Coordination Between Constellation Elements

APS has been extended through multiple programs for AFRL, DARPA, NASA, ONR, and commercial entities to include support for fully decentralized collaborative asset autonomous planning. This is facilitated through the sharing of a Common Relevant Operating Picture (CROP), essentially a distributed database, containing asset states (dynamic and related to onboard resources) and a representation of the mission-oriented "work items" that the swarm is being asked to perform. These work items align with a "workflow pipeline" defining the various steps that need to be undertaken to accomplish a high-level mission objective. The module in the APS architecture that handles the synchronization of this CROP between the assets in the system is called the Constellation Data Manager (CDM). It employs intelligent approaches to the distribution of data (namely the use of gossip protocols, value thresholding, relevance scoping, and compression techniques) to maintain the CROP data across the networked assets using minimal bandwidth.

Regardless of mission objective and resource differences between APS-enabled constellation assets, all assets and ground interfaces communicate to the APS constellation across a set of synchronization messages between instances CDM running on each deployed asset and ground station. The CROP representation is composed of chained pipelines of work items in which each work item represents an atomic portion of work to be performed in fulfilling a higher-level request. The synchronization messages containing work item details are then sent between CDM instances to disseminate high-level requests from users, notify the constellation of autonomous follow-up work items triggered by completed work items, and to relay work item state across the constellation. The combination of APS's mission flexibility features and the lightweight synchronization of work item state between CDM extend single-asset autonomous planning and scheduling to constellation-level plan optimization.

To minimize bandwidth requirements, we leverage an Event-Triggered Distributed Data Fusion (ET-DDF) algorithm to coordinate the CROP. The specifics of the algorithm are beyond the scope of this paper, but the central idea is that instead of constantly exchanging information – as one may do to share relative positions when using a standard approach such as a Kalman filter – only statistically relevant changes (the eponymous 'events') are shared. This greatly reduces the communication traffic required to effectively coordinate the CROP. We refer the interested reader to Ahmed et al.[7,8] for a more detailed discussion of ET-DDF.

Load Balancing Among Constellation Elements

The shared awareness of constellation capabilities gained via CROP data sharing is used by APS to coordinate activities across constellation elements. APS constellation coordination can manifest in several ways; we present two basic interaction methods here: bidding and handoffs. While these capabilities are foundational, they are straightforward and we note that there are more nuanced and complicated ways of coordinating actions across the constellation. A proper treatment of ways to coordinate actions[9] or design network topologies[10] is a big area of ongoing research in Multi-Agent Systems (MAS) and is thus beyond the scope of this paper. However, APS's flexibility allows any such interaction methodology to be implemented within the current architecture.

Bidding may occur when multiple constellation assets individually perform work item assessment and either bid on a work item based on estimate of the FOM to complete it, or standing down if the estimated FOM is lower that what any other asset has asserted. In this way, the multi-factor FOM associated with the work item is optimized to a maxima with vehicle capability and schedule conflicts in mind via this coordinated hill climbing technique. The bidding technique has innate resiliency to the single point failures that might occur in systems employing centralized planning and scheduling approaches, or when communication disruption occur. In the case where communication is disturbed between constellation assets, the worst-case result is over-

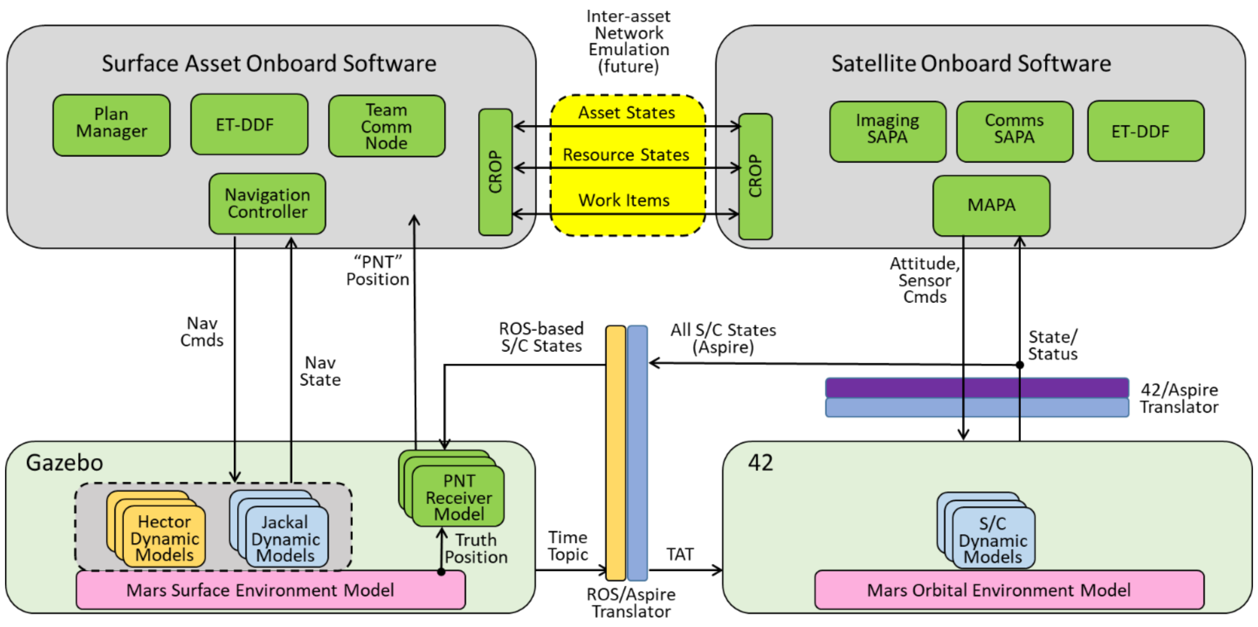

Figure 7. Block Diagram of Demonstration Simulation Environment.

satisfaction of work items by multiple constellation assets. True redundancy is also realized through this bidding approach, since inoperable constellation assets do not prevent work items from being fulfilled.

Work item handoffs occur when different work items in a pipeline are fulfilled by different constellation assets. Handoffs may occur in missions with advanced objectives and heterogenous constellations. For example, in a case where collected data might subsequently be processed on another capable vehicle, the data must be transferred. In these cases, SAPAs must track work item dependencies as they are planned and fulfilled in order to be able to plan downstream items in the pipeline. This technique may also be used in conjunction with bidding to provide further optimization and redundancy to more complex missions.

The greatest challenge to distributed TCPED architectures in general is the orchestration of data transfer between steps in the workflow pipeline given that some system configurations may have very dynamic or intermittent communication opportunities among certain assets. This necessitates the use of the delay-tolerant networking approaches and decentralized planning logic implemented in APS. In certain cases, APS will need to determine how to route data between system elements using multiple network hops.

The Mission Executive

The Mission Executive software manages the software modules comprising an APS deployment on a hosting platform, whether that be on the flight computer or on a dedicated co-processor. It is responsible for starting, stopping, and monitoring the APS application suite (e.g. MAPA, SAPA, other services). The Mission Executive makes use of various configurations settings to maintain flexibility over a variety of host platforms.

The Mission Executive can be broken down into the following functional components:

- **Process Control & Monitor.** This function entails Mission Process Control – the ability to start/stop APS-related processes and receive statistics on each– and Mission Process Monitoring – the ability to monitor process startup and health (via heartbeats).
- **Host System Interactions.** This component entails Message Interpretation – the ability to receive configuration/control commands from the hosting asset (includes settings changes and manual process controls) – and Telemetry Formatting – the production of telemetry packets containing Mission Executive and APS application module status.

The Mission Executive software is an "always running" process on the hosting Linux OS. The software is invoked by default (i.e. as a service) any time the host system is booted into a normal operating mode. Upon startup (or based on a user directive to restart the service), the Mission Executive software reads in the contents of the "Application Suite" defined in its Software Suite Specification file. This file serves as a configuration file for various necessary parameters, and

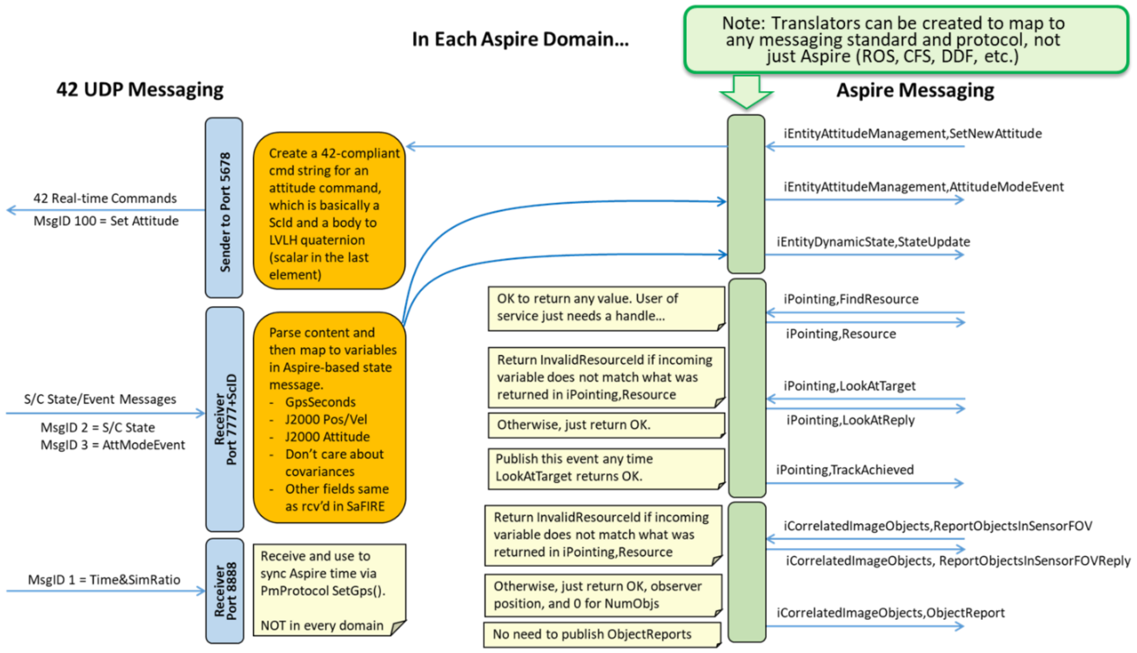
In Each Aspire Domain...
Note: Translators can be created to map to any messaging standard and protocol, not just Aspire (ROS, CFS, DDF, etc.)
42 UDP Messaging
Aspire Messaging
42 Real-time Commands
MsgID 100 = Set Attitude
Sender to Port 5678
Create a 42-compliant cmd string for an attitude command, which is basically a ScId and a body to LVLH quaternion (scalar in the last element)
iEntityAttitudeManagement,SetNewAttitude
iEntityAttitudeManagement,AttitudeModeEvent
iEntityDynamicState,StateUpdate
S/C State/Event Messages
MsgID 2 = S/C State
MsgID 3 = AttModeEvent
Receiver Port 7777+ScID
Parse content and then map to variables in Aspire-based state message.
- GpsSeconds
- J2000 Pos/Vel
- J2000 Attitude
- Don't care about covariances
- Other fields same as rcv'd in SaFIRE
OK to return any value. User of service just needs a handle...
iPointing,FindResource
iPointing,Resource
Return InvalidResourceId if incoming variable does not match what was returned in iPointing,Resource
Otherwise, just return OK.
iPointing,LookAtTarget
iPointing,LookAtReply
Publish this event any time LookAtTarget returns OK.
iPointing,TrackAchieved
MsgID 1 = Time&SimRatio
Receiver Port 8888
Receive and use to sync Aspire time via PmProtocol SetGps().
NOT in every domain
Return InvalidResourceId if incoming variable does not match what was returned in iPointing,Resource
iCorrelatedImageObjects,ReportObjectsInSensorFOV
iCorrelatedImageObjects, ReportObjectsInSensorFOVReply
Otherwise, just return OK, observer position, and 0 for NumObjs
iCorrelatedImageObjects,ObjectReport
No need to publish ObjectReports

Figure 8. Detailed mapping between Aspire/APS and 42 data protocols

contains a list of application entries to be controlled by the Mission Executive software. Since the Mission Executive functions outside of the Aspire middleware that allows plug-and-play interoperability between APS-related software modules, it maintains a "direct line" of communication to the host platform for command and telemetry messages. Once the APS Aspire applications are started, they can attach to the middleware and communicate application health and status data to the Mission Executive.

The Mission Executive starts the applications that comprise APS and monitors their health. Once an app is started, registered, and has successfully hooked all its message dependencies, the Mission Executive tracks its health using a heartbeat "ping-pong" to assess responsiveness. An app is considered nominal if a heartbeat response is received within a configurable timeout window.

An app that fails to respond soon enough (within a configurable time window) is considered "dead", and is subject to a series of possible failure responses. The failure responses are specified on a per-app basis as defined in the Software Suite Specification. Configured response behaviors may change based on specific mission needs, mission phase (i.e. commissioning vs. nominal flight) and other factors as dictated by the Operations Team. Potential responses include no action, stopping the app, restarting the app, stopping the entire APS application suite, restarting the entire suite, or resetting the application suite to a previous system state.

The Vehicle Interface Translator

To facilitate deployment on arbitrary platforms, we have developed a Vehicle Interface Translator to pass messages between internal and external messaging protocols. It describes all components designed to interface directly with the vehicle system in order to relay known vehicle state to APS, carry out APS planned activities, and provide a maintenance port for updating APS binaries and configuration.

As a concrete example, we consider a demonstration scenario in simulation, where APS must communicate with robotic and orbital simulation software suites, Gazebo and 42, respectively; see Figure 7. Figure 8 provides a detailed view of the mapping configured in the translator. The exchange of information between the 42 and Gazebo realms was facilitated by the configuration of a translator to map between the Robot Operating System (ROS) and Aspire messages supported by each of the simulation environments (ROS on the Gazebo side and Aspire on the 42/APS side). The translator was built using technology already matured in other AFRL-sponsored research efforts, where it plays a vital role in allowing modular middleware-based onboard software architectures to seamlessly interoperate with the legacy data protocols of the hosting platforms that autonomy software is installed on. Here it merely allows us to easily allow two middleware environments to communicate. Previous bridging/translation with our compatibility layer had addressed mapping Aspire messaging to custom mission protocols for AFRL, namely payload-to-bus protocols

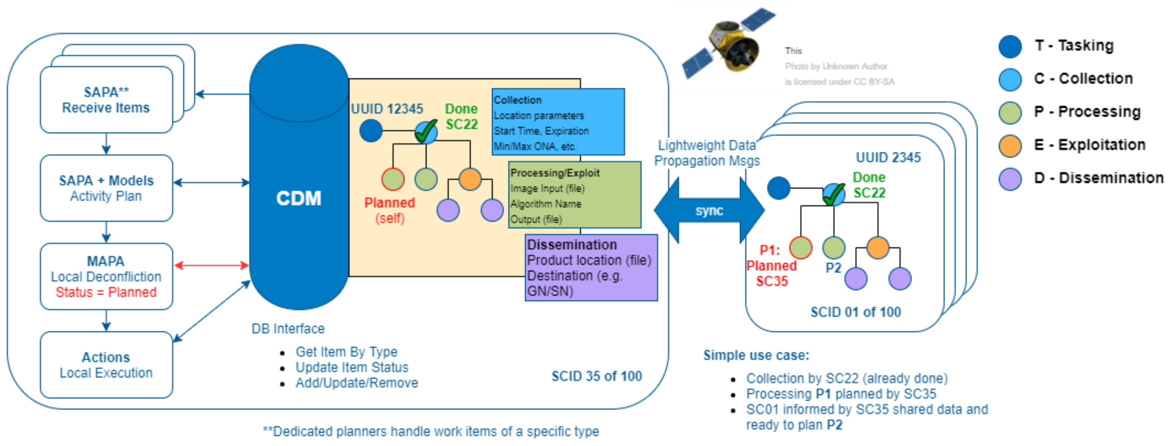

Figure 9. APS and the TCPED pipeline.

associated with their TacSat and EAGLE missions. These are custom packet protocols exchanged over point-to-point physical interconnects like SpaceWire or RS-422.

This translator is shown as the yellow/cyan sandwiched blocks at the middle bottom of Figure 7. The translator allows the time as modelled by Gazebo to be delivered to 42 and the truth states of the satellites, as modeled by the dynamics engine of the 42 simulator, to be delivered to the Gazebo-supported elements.

Hardware Details

APS has been deployed on Unibap e2100, Raspberry Pi computers, and other constrained platforms. It has been used on Raspberry Pi for testing in deployed unmanned underwater vehicle (UUVS) applications. It will fly on two satellite missions in 2021 and 2022. One employing an Innoflight CFC-400 and a the second employing a custom board utilizing a Xilinx Versal chip. APS runs on Linux- and Windows-based systems on 32- and 64-bit x86 and ARM hardware computing architectures with minimal package requirements and low memory and computing requirements, making it suitable for installation on resource constrained flight computers for satellites and various classes of unmanned vehicles.

APS AND THE TCPED PIPELINE

Our approach manages and orchestrates processing across all platforms in the architecture using a decentralized database that maintains representations of data processing pipelines, essentially a collection of relational graphs where the nodes are work items (the types of which represent steps in the TCPED process) and the vertices represent the dependencies between them (which steps need to be completed in order to move to the next step). Pipelines may be simple linear workflows (collect, process, disseminate), or complex (multiple collects of different phenomenologies, fuse via a processing step, use processing results to cue another type of collection, process that data to identify features, disseminate features to specific users). These pipelines include the status of each work item (whether it has been planned, is in progress, or is complete) as well as metadata associated with accomplishing the step (begin/end times, platform satisfying the step, and a multi-factor figure of merit score representing how well the work item will be satisfied). These pipeline representations are synchronized across all platforms using a gossip protocol that minimizes data exchange on the communication links.

This approach is layered-upon (and agnostic-to) the underlying physical layer that supports communication between assets. A decentralized planning suite on each platform consults the work pipelines to determine whether local resources can be utilized to satisfy various work items. If possible/feasible, an optimized plan will be created and an associated score posted within the work item. Other assets in the system may also develop plans, but will stand down if their scores are lower. The result is a fully decentralized self-selection of which steps in the pipeline will be satisfied by which platforms. This approach is ideal for highly dynamic systems of systems that are handling large numbers of user data requests of differing priorities from a large community of users.

DEMONSTRATION SCENARIO

To illustrate APS in operation, we present a demonstration scenario from our MISDEF program, initially described in the Section, Motivating Scenario: Collaborative Robotic Exploration of Mars. The scenario for which we will show results concerns heterogeneous swarms of satellites, rovers, and atmospheric vehicles.

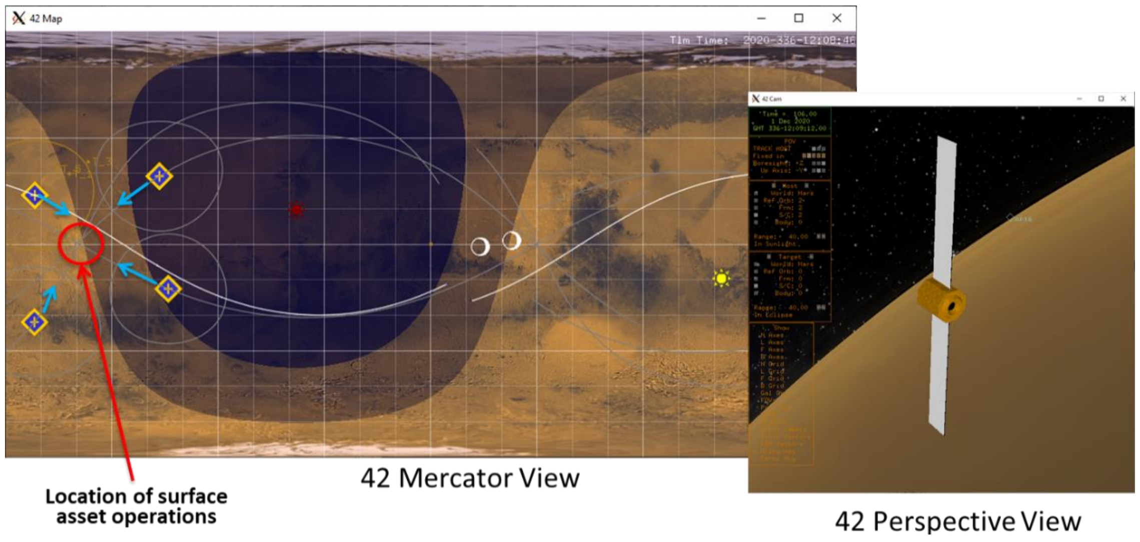
42 Map
Sim Time: 2020-336-12:08:48
Location of surface asset operations
42 Mercator View
42 Perspective View

Figure 10. 42 screenshots of the satellites' converging orbits over the surface assets' operating area

Figure 7 shows a block diagram of the simulation, which includes Gazebo for modelling robot dynamics and the 42 simulation framework[12] for modeling the orbital environment.

Gazebo acts as the simulation time reference since it has no native ability to be driven by an external time source. Thus, 42 references off Gazebo's time. We accomplished the synchronization capability by extending the 42 message set to be capable of receiving a run-time message setting the desired "current" time. On the ROS side, Gazebo publishes a Time topic that was usable to assert the "wall-clock" time as it executed.

The demo scenario involves four satellites in orbits (300km altitude at a variety of inclinations) that result in them all converging over a specific ground area at roughly the same time. The satellites host Positioning, Navigation, and Timing (PNT) transmitters that deliver signals toward nadir as they traverse their respective ground tracks. A PNT receiver installed on any surface or atmospheric asset is able to determine its absolute position to a level of fidelity that depends on the number of satellite signals being obtained and the relative line of sight geometries to the transmitting satellites. When multiple satellites are in view of any surface asset, that asset is able to have knowledge of a "GPS-like" position fix. When not providing PNT service over the operational theater, the satellite assets would autonomously plan and execute orbit to surface image collection.

Demo Scenario Description

The Demo brings together a relevant Mars heterogeneous asset decentralized planning, orchestration, and execution capability leveraging the team's collective capabilities. The robotic team is entrusted with an over-arching high-level objective – to identify potential areas of interest (AOIs) within a large Mars surface region and perform successive stages of further inspection/follow-up using assets with varying capability, as shown in the left graphic of Figure 13. This top-level goal includes specification of a large region surrounding the location of the surface team (we used a rectangular latitude/longitude bounded region of 20x20km). This objective could be issued by an Earth-resident mission operator, or it could alternatively have

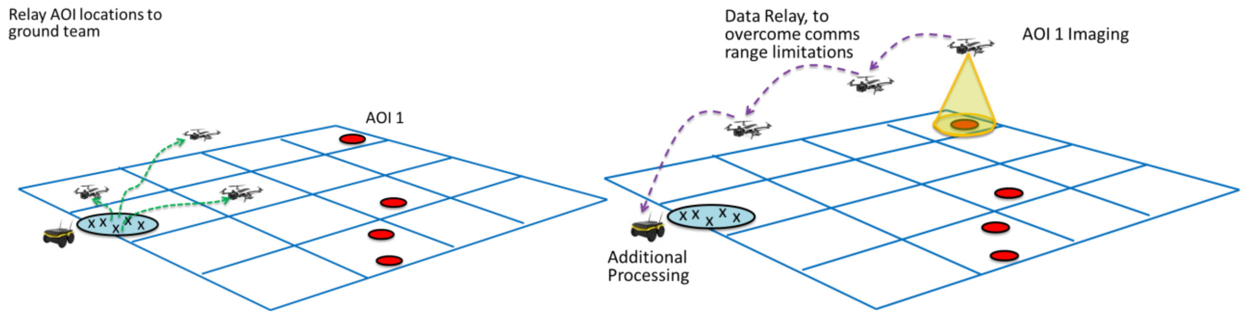
Relay AOI locations to ground team
AOI 1
Data Relay, to overcome comms range limitations
AOI 1 Imaging
Additional Processing

Figure 11. MRACC algorithm determining drone sortie to image AOI 1 and return data.

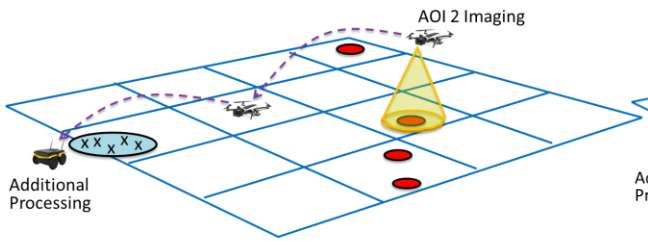

Figure 12. MRACC servicing successive AOIs.

been generated autonomously by some mission decision logic (out of scope of this demo).

Orbital Assets

The satellite assets, being in possession of sensors capable of acquiring tiled imagery of the region, receive the center points of sub-regions that have been decomposed from the full region specification. Each satellite's APS planning software determines access opportunities to image each sub-region, as well as a goodness score. Those scores are shared between all satellites that might perform the collections. The satellite with the best score ends up committing to the image acquisition activity, while others with lower scores stand down. When a satellite performs its collections they simulate passing those images through a detection algorithm to reveal possible AOIs. Any AOIs discovered are transmitted in an event message to the rover acting as the surface activity coordinator (base station).

Atmospheric Vehicles

When the rover orchestrating surface activity detects, using PNT signals, that the satellites have exited the surface theater, it accumulates all AOIs into a single message and provides it to the MRACC algorithm running decentralized on all atmospheric vehicle drones. MRACC then orchestrates the dispatch and navigation of the drone team to service each AOI in an order based on the over-surface path distance and the AOI's associated priority. "Servicing" each AOI means that a drone hovers over the location and acquires imagery, which it relays to the base station through a "chain" of drones (because of range-based communication limitations).

Each of these acquisition/relay activities is conducted by the drone team until all AOIs have been serviced, at which point the sortie concludes and all drones return back to the base station location.

Rover Operations

As soon as the data for the last AOI acquisition is returned by the final relay configuration, software on the base station processes the acquired data and looks to see if there are indications that follow-up by a rover might be appropriate, e.g., to perform contact surface science. In this demo scenario, one AOI is deemed worthy of follow up. Decision logic on the base station makes the determination that a certain rover should perform the follow-up activity. For our demo we simply made that determination based upon proximity to the AOI's center – closet rover is favored to make the excursion. That instruction (to navigate to the AOI) is published on the message bus and acted upon by the chosen rover. Once the rover is en-route, the demo concludes.

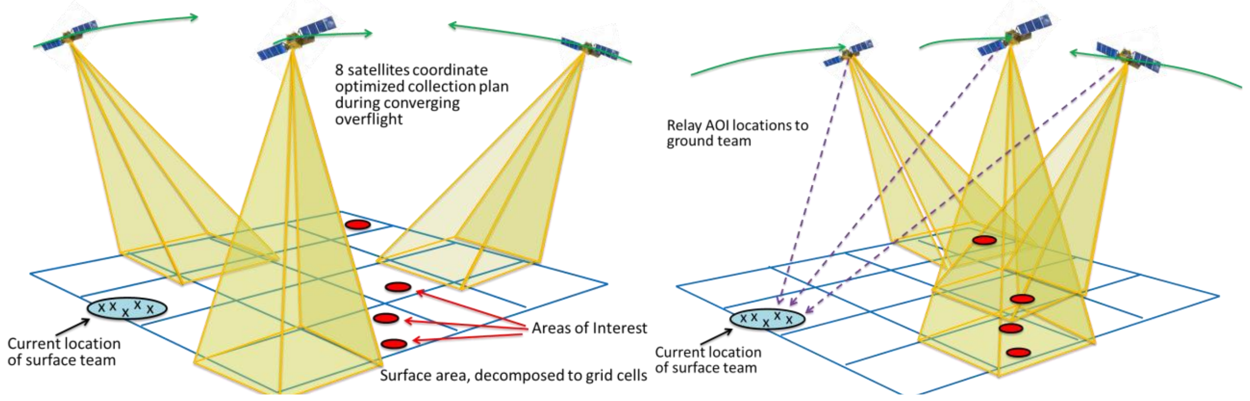
8 satellites coordinate optimized collection plan during converging overflight
Current location of surface team
Areas of Interest
Surface area, decomposed to grid cells
Relay AOI locations to ground team
Current location of surface team

Figure 13. Satellites coordinating and executing image collection and processing to reveal AOIs.

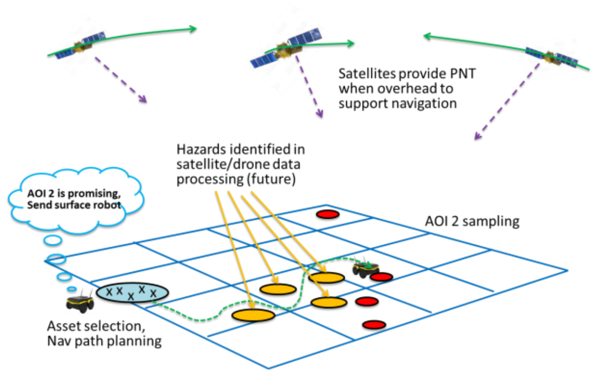
Satellites provide PNT when overhead to support navigation
Hazards identified in satellite/drone data processing (future)
AOI 2 is promising, Send surface robot
AOI 2 sampling
Asset selection, Nav path planning

Figure 14. Follow-up with contact science by rover.

Details on Interactions Between Platforms

Figure 15 depicts key interactions between modules involved in the demo. As previously mentioned, all activities are directed in decentralized fashion by the collaborative team once the initial “Directive” is introduced (also called a Mission Data Request, or MDR). We used one of the tools accompanying AFRL’s Aspire middleware (called the Aspire Studio browser) to provide the initial message (indicated by #1 on the figure). We created the Mission Objective Manager (or MOM) module to field the MDR and decompose it into a set of “work items” aligned with steps in the TCPED process (the life cycle of delivering a final end data product to a user or users resulting from the tasking of sensors). The MOM decomposes the top-level MDR into multiple collection, processing and dissemination tasks. Each of these is pushed into a workflow specification held by the Ground Target Manager component (leveraged from other programs with AFRL and DARPA). At this point, all assets know that the work items exist and are in need of being planned.

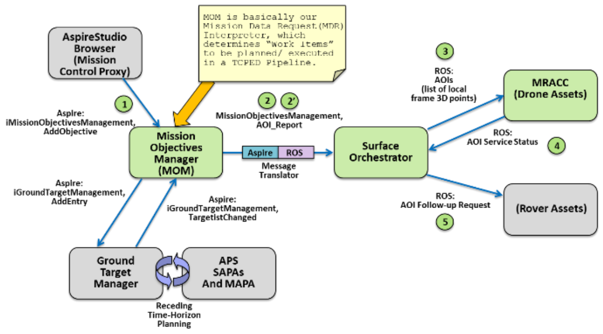
MOM is basically our Mission Data Request (MDR) Interpreter, which determines "Work Items" to be planned/ executed in a TCPED Pipeline.
AspireStudio Browser (Mission Control Proxy)
1
Aspire: iMissionObjectivesManagement, AddObjective
Mission Objectives Manager (MOM)
2 2'
MissionObjectivesManagement, AOI_Report
Aspire ROS
Message Translator
Surface Orchestrator
3
ROS: AOIs (list of local frame 3D points)
MRACC (Drone Assets)
ROS: AOI Service Status
4
ROS: AOI Follow-up Request
5
(Rover Assets)
Aspire: iGroundTargetManagement, AddEntry
Aspire: iGroundTargetManagement, TargetlstChanged
Ground Target Manager
APS SAPAs And MAPA
Receding Time-Horizon Planning

Figure 15. Module interaction diagram supporting Demo capabilities.

The APS SAPAs on each satellite consider each work item and plan the time at which they can occur (if possible), scoring each and pushing that score to the Common Relevant Operating Picture (CROP) decentralized database. As previously mentioned, all assets will use those scored to either self-select or stand-down on each work item. Plan status is also held in the CROP, so any module with access to the CROP interfaces (which includes the MOM) will be aware of the fact that work items have been planned, and aware of each work item’s completion.

Once the MOM recognizes all work items as having been completed (satisfactorily or not, for the current satellite fly-over) any resulting AOIs detections are bundled and delivered to the module called the Surface Orchestrator, which lives on the ROS side of the simulation architecture and is presumed to run on a rover designated as acting in the role of the base station (most likely because it hosts the best surface-orbit communication equipment, or perhaps because it has the greatest processing/memory resource capacity of any of the surface team).

As previously described, the Surface Orchestrator will perform the delivery of messages for AOI exploration by the drone team, and will also apply simple (at this point in the research) decision logic to select a rover for contact follow-up of any AOIs that have the greatest likelihood of high science return using certain rover-hosted instruments.

Mars Robotic Asset Cooperative Control (MRACC)

In the demo, Mars Robotic Asset Cooperative Control (MRACC) coordinates multiple air and ground assets with limited communication range to overcome data delivery issues using a multi-hop communication scheme.

The MRACC algorithm is a distributed optimization algorithm based upon the Communication-Aware Robotic Information Gathering (CARING) framework developed by the University of Colorado[5]. Figure 16 shows the block diagram that illustrates how MRACC operates between communication and platform. Firstly, MRACC receives a data package by communication status. The package consists of the current ET-DDF estimates of quadrotors over multi-hop communication from the base station, decision set of other quadrotors, and science data. If a quadrotor disconnects to other quadrotors, then the quadrotor uses data packages that are received recently. Then MRACC predicts future positions of quadrotor assets through delivered decisions of other quadrotors with higher ranks in a hierarchy and one of the quadrotor's discrete decisions. Next, MRACC optimization performs using a local utility formulated as a sum of all possible values of which element refers a specific communication event. The value is computed by multiplication between a probability of delivery and information gain given the configuration (Figure 17).

This demo considered the information gain as a function of the relative distance between a selected AOI and quadrotor position estimates, where the information gain increases as the quadrotor approaches to the AOI. Finally, MRACC computes local utilities for variation of decisions itself and takes one of them that maximizes the local utility. The optimization recurs periodically.

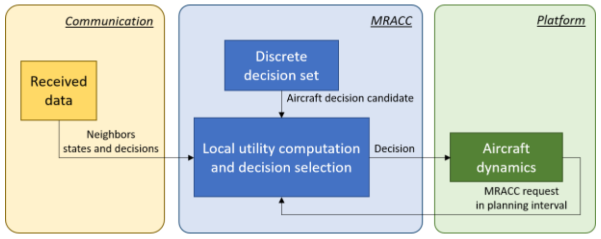
Communication
Received data
Neighbors states and decisions
MRACC
Discrete decision set
Aircraft decision candidate
Local utility computation and decision selection
Decision
Platform
Aircraft dynamics
MRACC request in planning interval

Figure 16. Block diagram of MRACC. MRACC uses received data over communication and given discrete decision-set to compute local utilities and select decisions that maximize local utility. The quadrotor moves based on the decision and requests MRACC when planning interval time elapses.

Due to imperfect communication, MRACC works in a distributed way, where each quadrotor locally takes MRACC to make its decisions. One way to cooperate/coordinate distributed robotic systems is to receive the decision data from other quadrotors based on the rank in a hierarchy. Other quadrotors' decisions are transmitted with sensor data and current estimates served by base station over communication. Note that the decisions may not be delivered when two quadrotors are disconnected. In that case, the quadrotor ignores the decision even though the disconnected quadrotors have a higher rank in a hierarchy.

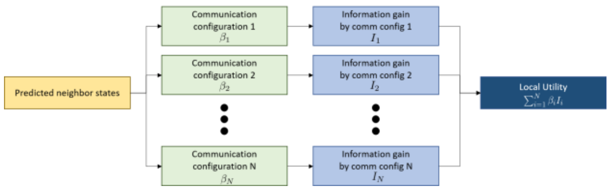
Predicted neighbor states
Communication configuration 1 β_1
Communication configuration 2 β_2
Communication configuration N β_N
Information gain by comm config 1 I_1
Information gain by comm config 2 I_2
Information gain by comm config N I_N
Local Utility $\sum_{i=1}^{N} \beta_i I_i$

Figure 17. Black diagram of MRACC local utility computation. MRACC considers N cases of multi-hop communication events of quadrotor assets and a base station where each event has its information gain. MRACC computes local utility by taking the sum of the product between the probability of communication configuration and its information gain, which results in the expected communication-aware information gain given predicted quadrotor states.

Simulation Results

Figure 18 shows screenshots of visualization obtained during a run for the satellite operation component, which involves a 12 satellite constellation (3 orbit planes with 4 satellites equally spaced within the plane). Their 1000km orbit gives them a wide field of regard, allowing the imagers mounted on the agile bus platform to be commanded to acquire ground targets from a collection of 500, located in 7 regional clusters scattered around the Martian surface.

APS's decentralized collaborative planning uses plan and score sharing to achieve collection of images of the best quality within the temporal and lighting constraints associated with each order. Proper targeting of the ground targets by the 42-hosted satellite models an erosion of all targets in the mission request queue within the defined mission execution window was verified.

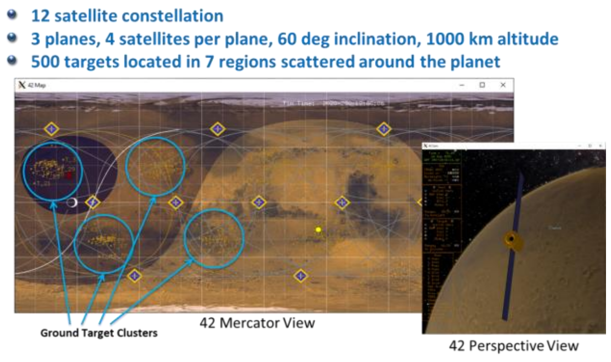
12 satellite constellation
3 planes, 4 satellites per plane, 60 deg inclination, 1000 km altitude
500 targets located in 7 regions scattered around the planet
Ground Target Clusters
42 Mercator View
42 Perspective View

Figure 18. Visualization Views of Scenario 2, Mars-wide image collection

The scenario for verification of MRACC was as follows. This demo's goal was to relay streamed sensor data back to the base station during collects using quadrotor assets. The base station would then dispatch the nearest ground vehicle to the selected AOI based on the data to perform follow-up science missions based on the collected data by the quadrotors. Three quadrotors and four ground rovers were initially at the base station (Figure 19). The quadrotors team started to achieve missions that provide hovering data collection of three stationary AOIs, where the simulation located the AOIs within 1500 meters of the base station. The quadrotors sequentially selected one of AOIs (i.e., AOI 1→AOI 2→AOI 3) and figured out the optimal positioning with communication boundary (<500 meters) probabilistically determined by hardware parameters given in Campbell and Ahmed[13] and Ahmed[14]. The simulation used the multi-hop communication, in which multiple data relays from the nearest quadrotor that obtained science data of an AOI to the base station happened. The communication modeled as a packet erasure channel, where the outcome of communication for single-hop was *delivered* or *not delivered*.

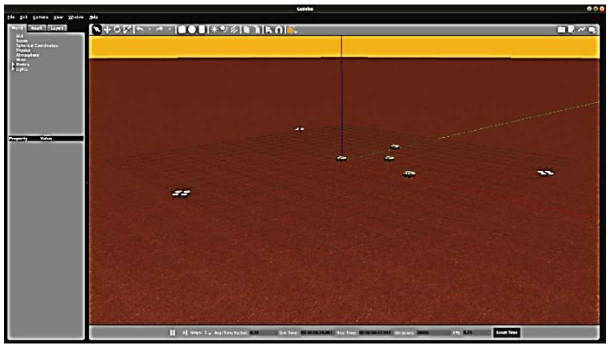

Figure 19. Screenshot of Gazebo with three quadrotors and four ground vehicles.

Figure 20 shows the trajectory using MRACC. This demo used three quadrotors (Hectors) and one base station (Jackal) at the origin where the mission planner sequentially selected one of three AOIs for science missions. The relative distance between one Hector to another/base station determined the probability of delivery over single-hop communication. As referred by Campbell and Ahmed[13] and Ahmed[14], the communication successfully happened within a 500-meter range, and the probability of delivery drastically dropped when the distance was between 500 meters and 600 meters. No communication showed over 600 meters.

First, three Hectors took off and headed to AOI 1, the first selected AOI for sequential science missions. When the distance between the Hectors and base station was over 500 meters, Hector 1 took communication relay. Hector 2 and Hector 3 could collect science data of AOI 1, in which Hector 2 collected, and Hector 3 took a back-up position to recover when Hector 2 failed. After taking the science mission for AOI 1, all three Hectors moved toward AOI 2, where Hector 1 took the relay role to deliver data from other Hectors to the base station. Because one Hector was not enough to cover the multi-hop communication range when collecting data of AOI 2, Hector 2 took the other relay position to deliver data obtained by Hector 3 to Hector 1. When Hectors finished the search mission for AOI 2, three Hectors moved to AOI 3. Hector 2, the nearest asset to AOI 3, collected data and Hector 1 relayed the generated data to the base station, which Hector 3 acted as a back-up of Hector 2. All three Hectors returned to the base station after finishing data collection for all three AOIs.

Note, MRACC did not use any task allocation for data collecting/relaying. The MRACC showed the high fidelity of communication coverage and the mission achievement in a distributed manner. Furthermore, MRACC performed in real-time, so no pre-planning procedures were required.

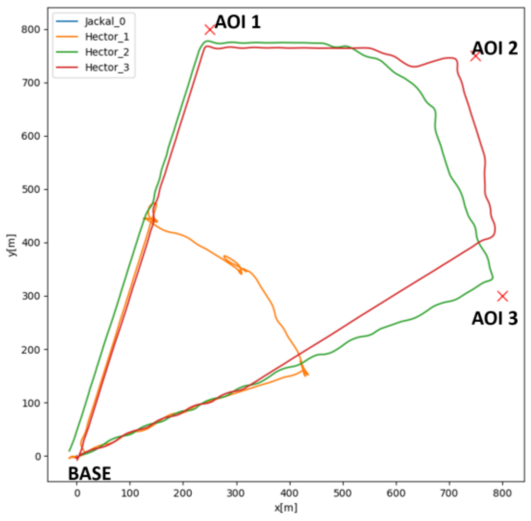

Figure 20. MRACC sample result for coordination of three rovers with three AOIs.

DEPLOYMENTS OF APS

In this section, we discuss several domains for which APS is relevant. A summary of resources for which APS components has been developed can be found in Table 1. We note that this list is not exhaustive and that developing additional components can be simple due to the flexible and powerful architecture underlying APS.

APS on Different Platforms

The integration of APS onto a host platform is designed to be simple and flexible. Though APS natively uses the lightweight Aspire middleware messaging framework for its own inter-modular data interactions, it also provides flexible translation to and from external data interfaces through a Vehicle Interface Translator component that supports customized plugins to meet mission-specific formats and protocols. As APS plans are generated and changed, an action interface may be used by the vehicle to drive a vehicle execution timeliner component to carry out the results of planning. These actions are also broken down in several actionable levels for use by the vehicle based on controller capabilities. For example, a maneuverable spacecraft may need to perform a point/hold action to stare a sensor boresight at a point on Earth's surface. This point/hold action is also broken down into more digestible actions as a set of slew and spin commanding. The customer may then decide which of these action abstractions works best to carry out the actuation based on the onboard attitude control system and its level of commanding and behaviors.

Table 1. Multi-Domain Resources Supported by APS

Resource	*Domain*	*Description*
Attitude Control	Satellite Operations	Supports operations to change the orientation of the satellite to support different operations; e.g., slewing the satellite body to image with a fixed sensor
Satellite Maneuvering	Satellite Operations	Supports operations to change satellite orbit, e.g., to avoid a potential collision
Optical/RF/SAR/HS Imagers	Satellite Operations, Robotic Exploration	Supports operations to collect data with different imaging sensors; this includes slewing, recorder management, etc.
Wheeled Locomotion	Robotic Exploration	Supports navigation of wheeled vehicles such as rovers
Rotorcraft Locomotion	Robotic Exploration	Supports navigation of rotorcraft vehicles such as UAVs
Electrical Power	All	Supports electrical power management and modeling for activities executed on any platform; configurable modeling and estimation considers vehicle position and orientation, solar panels, and power draws incurred through vehicle actions through the rolling time horizon.
Data Recorder	All	Supports data capacity estimation and management on board based on data generated by science activity collection and deletion behavior after data has been successfully downlinked
Communication and Contact Resources	All	Supports communication device modeling, estimation, and deconfliction used for contact scheduling between the ownship vehicle and any other multi-domain asset. Communication modeling models the radio and antenna capabilities to optimize contact activities between constellation assets

Orbit Logic's core ground-based software products (Collection Planning and Analysis Workstation (CPAW), Order Logic, and SpyMeSat) have been integrated with APS to provide more powerful solutions. This integration allows for a mission to blend the computational power of the ground-based solutions with the low Size, Weight, and Power (SWaP) and urgent tasking capabilities of APS. For example, integration with Orbit Logic CPAW may be used by an operator to orchestrate, tweak, and improve a constellation collection plan and then upload that plan for execution by an APS-enabled constellation. Individual constellation assets will execute the CPAW plan, but will subsequently field urgent (high priority) requests and attempt to insert them into their execution schedules.

Satellite Deployments of APS

APS was originally developed for AFRL for satellite operational resiliency, self-protection and the enhancement of local Space Situational Awareness (SSA). The use cases addressed include monitoring the local space environment around the satellite to maintain state awareness of known proximal objects, to detect and characterize new objects, and potentially to react in real-time to these events. An Intelligent Search SAPA determines optimized search patterns based on user-specified watch volumes and volumes generated by fusion algorithms associated with probabilistic regions of object reachable orbital states. The MAPA generates a de-conflicted execution schedule for the use of local satellite resources (SSA camera, satellite attitude control, communications, and thrusters) to achieve competing observation and search objectives without violating satellite/sensor keep-out constraints or over-utilizing spacecraft resources

For DARPA's Blackjack program, APS was used to perform both satellite constellation- and asset-level planning, employing a fully-decentralized approach involving only the minimal exchange of assets states and plan FOM scores and status. In scaled-up testing against mission simulations, APS's receding timeline approach proved capable of effectively developing coordinated plans for the collection of thousands of user-specified ground targets by multiple sensor phenomenology types (EO, IR, RF) hosted on hundreds of satellites.

APS is flying on a Loft Orbital hosted satellite demonstrator mission for the DARPA Blackjack program in June of 2021. The mission (focused on maritime domain awareness) will demonstrate the collaborative planning elements of the modular Blackjack system design by participating as a live element in a ground-based Live-Virtual-Constructive (LVC) simulation. Part of that experiment's purpose is to demonstrate on-orbit reconfiguration/update of APS software.

APS will also fly on a university operated satellite in 2022. This partnership will mature APS and provide a platform for the university to test new research and capabilities in a real-world setting.

Unmanned Vehicle Swarm Deployments of APS

In addition to the MISDEF effort partially described in this paper, Orbit Logic is teamed with the University of Colorado Boulder on the Intelligent Navigation, Planning, and Autonomy for Swarm Systems (IN-PASS) solution for NASA to address the coordination of autonomous robotic exploration activity on the Moon. IN-PASS uses multi-objective control policies to tradeoff rover navigation performance and resource use. It is also investigating the most effective means of interaction between humans and the swarm elements, including specification of goals/objectives, feedback on the viability/status of the human's requests, and the delivery of the resulting science product back to the human.

On the MinAu program with the US Navy, Orbit Logic is partnered with the University of Colorado Boulder and the University of California San Diego, and NIWC-PAC. APS is integrated with (and validated against) NIWC's hardware systems, which uses SeaRover UUVs - tetherless versions of the BlueROV2 platform that host a range of mission-relevant sensors and employ a USBL system to perform locatization and facilitate data exchange between assets. The missions investigated to date have included collaborative bottom mapping and water volume defense using collaborative search/detect/track behaviors.

6. Conclusions

Orbit Logic's Autonomous Planning System (APS) is onboard software that enables collaborative autonomy among multiple, heterogeneous assets. At the asset-level, it enables independent operation so that spacecraft can operate independently of a ground station, reducing latency between on-orbit events and responses and eliminating the ground station as a single point-of-failure. At the swarm- or constellation-level, it enables heterogeneous groups of assets to coordinate their actions in order to better perform their missions; e.g., one satellite may cue another on a collection it cannot perform due to orbital geometry or sensor payload. APS is flexible; it can and has been expanded to support novel capabilities and it can and has been deployed on diverse hosting platforms.

APS has been developed to support satellite constellations in a wide variety of operational concepts. In addition, it is being developed to coordinate robotic exploration missions with NASA, and maritime missions with the US Navy.

Acknowledgments

We would like to acknowledge our collaborators on different deployments of APS. They include Drs. Nisar Ahmed, Eric Frew and Morteza Lahijanian of the University of Colorado Boulder, Dr. Sonia Martinez at the University of California San Diego, and all of their research groups as well as Dr. Anthony Jones at NIWC-PAC's Unmanned Maritime Vehicles Lab.

References

1. J. Reilly, "Autonomous Operations for Responsive Spacecraft", Responsive Space Conference, 2006.
2. S. Chien, et al, "Improving the Operations of the Earth Observing One Mission via Automated Mission Planning", SpaceOps, Huntsville, Alabama, April 2010.
3. J. Frank, D. Iverson, C. Knight, S. Narasimhan, K. Swanson, M. Scott, M. Windrem, K. Pohlkamp, J. Mauldin, K. McGuire, H. Moses. Demonstrating Autonomous Mission Operations Onboard the International Space Station. Proceedings of the AIAA Conference on Space Operations, September 2015.
4. G. Aaseng, J. Frank, M. Iatauro, C. Knight, R. Levinson, John Ossenfort, M. Scott, A. Sweet, J. Csank, J. Soeder, A. Loveless, D, Carrejo, T. Ngo, Z. Greenwood. Development and Testing of a Vehicle Management System for Autonomous Spacecraft Habitat Operations. Proceedings of the AIAA Space Conference, 2018.
5. S. Moon and E.W. Frew, 2019. "A communication-aware mutual information measure for distributed autonomous robotic information gathering." IEEE Robotics and Automation Letters, 4(4), pp.3137-3144.
6. Herz, Ella, Doug George, Timothy Esposito, and Kenneth Center. "Onboard autonomous planning system." In SpaceOps 2014 Conference, p. 1783. 2014.
7. N. Ahmed, J. Cortes and S. Martinez, "Distributed Control and Estimation of Robotic Vehicle Networks: Overview of the Special Issue," in IEEE Control Systems Magazine, vol. 36, no. 2, pp. 36-40, April 2016, doi: 10.1109/MCS.2015.2512030.
8. Nisar R. Ahmed, Jorge Cortes, Sonia Martinez, "Distributed Control and Estimation of Robotic Vehicle Networks: An Overview of Part 2", Control Systems IEEE, vol. 36, no. 4, pp. 18-21, 2016.
9. Dorri, A., Kanhere, S.S. and Jurdak, R., 2018. Multi-agent systems: A survey. Ieee Access, 6, pp.28573-28593.
10. Jovanović, M.R. and Dhingra, N.K., 2016. Controller architectures: Tradeoffs between performance and structure. European Journal of Control, 30, pp.76-91.

11. IAF (International Astronautical Federation), 70th International Astronautical Congress, Washington D.C., United States, Monday, 21 October 2019.

12. N. Dhingra, K. Center, E. Herz, E. Sneath, S. Gagnard, “APS: Multi-Domain Decentralized Planning for Responsive Multi-Asset Collaborative Autonomy”, SpaceOps 2021.

13. E. Stoneking, “42: A General-Purpose Spacecraft Simulation”, NASA Software Designation GSC-16720-1,2010-2019. https://sourceforge.net/projects/fortytwospacecraftsimulation, https://github.com/ericstoneking/42

14. M. Campbell and N. Ahmed, “Distributed Data Fusion: Neighbors, Rumors, and the Art of Collective Knowledge,” IEEE Control Systems, vol 36, no. 4, 83-109, 2016.

15. N. Ahmed, “What’s One Mixture Divided by Another?: A Unified Approach to High-fidelity Distributed Data Fusion with Mixture Models,” 2015 IEEE Conference on Multisensor Fusion and Information Integration (MFI 2015), San Diego, CA.