SSC21-I-13

Constellation Modeling, Performance Prediction, and Operations Management for the Spire Constellation

Jeroen Cappaert, Frantisek Foston, Pablo Sierra Heras, Barry King, Nick Pascucci, Jordan Reilly, Conor Brown, Joey Pitzo, Marcus Tallhamn
Spire Global, Inc
8000 Towers Crescent Dr, Vienna, VA 22182; (202) 301-5127
jeroen@spire.com

ABSTRACT

The operational complexity of managing the Spire constellation continually increases with the routine introduction of additional satellites and new capabilities. The heterogeneous nature of the satellites, payloads, and ground station configurations compounds the difficulty of strategic planning and operational scheduling. In order to efficiently operate this diverse network of assets, Spire developed a suite of bespoke constellation modeling and management tools that are designed to support existing demand and to scale for future needs. The modeling tools enable Spire to accurately simulate and optimize the performance of various constellation configurations prior to deployment. The operational tools required to harness the full potential of the constellation incorporate complex techniques in order to schedule payload operations, maximize data collection, and monitor performance. These tools are developed in a modular and scalable fashion to ensure that new capabilities, such as the introduction of inter-satellite links, can be readily integrated into the planning system. In addition to these internal tools, Spire also offers a suite of standardized APIs and user services through which both internal and external customers can seamlessly integrate payloads and software with the Spire constellation, enabling secure access to development and simulation environments, scheduling, and data pipeline tools. The constellation modeling, performance prediction, and operational management tools developed at Spire are essential to ensure efficient and optimized production in an increasingly complex system.

INTRODUCTION

As Spire's smallsat constellation grows and more capabilities are added to the satellites, operationally managing the system becomes an increasingly complex task. At the current scale of the constellation, satellites can no longer be treated individually, but must be planned for as a strongly coupled system. Tasking space and ground assets, validating performance, and managing limited resources requires implementation at the systems level. With hundreds of satellites, dozens of ground stations, multiple payloads per satellite, and the use of inter-satellite links (ISLs), the scale of systems-level planning is immense.

The scope of the problem is compounded by the heterogeneous nature of the assets involved. The constellation consists of various generations of satellites with differing capabilities and constraints, while Spire's ground station network employs mainly owned capacity, but can also use leased capacity across multiple providers with varying performance characteristics. Current satellites are in multiple rideshare-based orbits, which causes changes in the data collection and communication availability on a daily basis. To manage a constellation of over 110 satellites and approximately 30 ground stations, Spire has designed and built a suite of constellation modeling and management tools.

The constellation modeling tool assesses the performance of potential future constellation configurations. This tool models satellite orbits, evaluates ground station placement, computes visibility between all assets, schedules communications, and determines figures of merit for the Earth observation data collected by the constellation. This information is synthesized to determine the return on investment for launches, ground station deployments, hardware and software features, and other critical concepts of operations.

Spire's operational planning tool, 'the Optimizer,' manages space and ground assets tasking in such a way as to maximize the business value of the fleet given a variety of operational considerations using a mixed-integer programming (MIP) model. Operational performance of payload collections and radio contacts are used to calculate value, and constraints in the model include limited resources on the satellites, such as power, available payloads, etc. Communication availability and contention between satellites and ground stations is also a key input to the tool.

To close the feedback loop, Spire monitors a variety of metrics that measure constellation performance and compares them with the modeled performance. These metrics monitor a variety of critical components of the system. Deviations from nominal values trigger automated alerts that are sent out to relevant working groups. Spire has also developed the 'Spire Operations Center' application which provides detailed monitoring of each asset in the constellation. Any identified discrepancies are then analyzed and corrected.

With the use of these tools and standardized constellation management system, Spire serves a large number of internal and external customers. Customers are provided an operational performance prediction before launch. During operations, customers may have Spire operate their assets as a service or use the operational planning and tasking tools themselves. Customers can also rely on quality performance through Spire's continuous monitoring of operations.

CONSTELLATION MODELING AND PERFORMANCE PREDICTION

Spire makes extensive use of constellation modeling in order to accurately predict and assess the performance characteristics of different constellation designs or configurations. At Spire, a constellation refers not only to the satellites and their orbits, but also to the ground station network and concept of operations. Modeling-derived performance data is a key input into the constellation design decision-making process (e.g., orbit selection, location of the ground stations, antenna definition parameters, etc.) and provides insights into the various trade-offs that may exist. This data-driven approach helps Spire ensure that the constellation will be able to meet the targets imposed by an increasingly diverse set of customers.

Modeling is used to translate business requirements into engineering requirements. Every customer imposes a unique set of requirements based on the characteristics of the data that must be produced by the Spire constellation. The requirements for the data collected from the sensors (the payloads) are usually quantified using the following metrics: *latency*, *revisit*, *probability of detection*, *refresh*, and *timeliness*.

Latency is the time delay from collecting data on a satellite to that data being available to a customer through one of the Spire application programming interfaces (APIs).

Revisit is the time between two consecutive observations by the constellation of a specific point on Earth.

Probability of Detection is how likely a satellite is to successfully collect the desired data while observing a specific point on Earth.

Refresh is the combination of revisit and probability of detection (i.e., the time between successful observations of a given target).

Timeliness is the combination of refresh and latency (i.e., how old is the latest message from a given target in the API). A boundary value for the maximum timeliness is the maximum refresh plus the maximum latency.

These concepts are further illustrated in Figure 1. An example of a customer requirement is the International Civil Aviation Organization (ICAO) 4D/15 requirement for air traffic monitoring. To comply with this standard, the operator must obtain four-dimensional aircraft position information (latitude, longitude, altitude, and time) at 15-minute intervals or less, which is a timeliness requirement that can be translated into latency and refresh requirements.

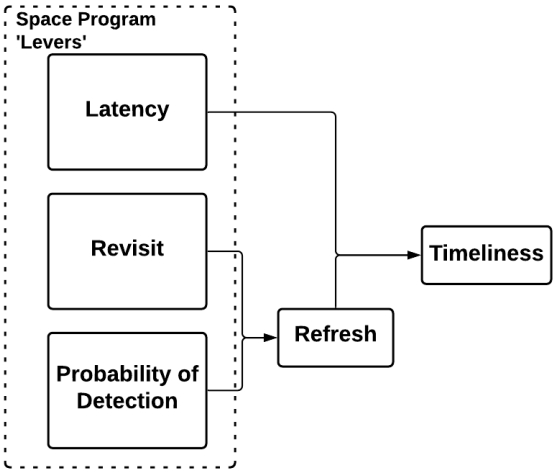

Figure 1: Metrics for measuring on-orbit data collection.

Inputs and Constraints

Details about the existing and future satellites and ground stations are required in order to define a candidate constellation.

For the satellites it is necessary to define the two-line element set (TLE), license country, hardware version, payloads, communications configurations, and scheduling constraints. The communications configurations define the type (i.e., ISL: satellite to satellite or GS: satellite to ground station), geometric

requirements, regulatory constraints, and transmission rates.

Payload operations determine the quantity and quality of data collected by a satellite. Spire's satellites are multi-sensor and can collect Automatic Identification System (AIS) messages from maritime vessels, Automatic Dependent Surveillance-Broadcast (ADS-B) messages from aircraft, and a range of Global Navigation Satellite System (GNSS) measurements, including GNSS-Radio Occultation (GNSS-RO) and Reflectometry (GNSS-R). Each payload definition includes a footprint shape, a fixed or variable data collection rate (bits collected per unit of time), the priority of the data collected by the payload, the probability of detection, and the scheduling constraints (e.g., duty cycle, maximum and minimum operational times, conflicting payloads, etc.).

For the ground stations, the location needs to be specified along with the communication configuration and scheduling constraints.

Customers must provide objectives for the constellation in order to evaluate the performance variances between different proposed configurations. These goals contain the areas of interest, the revisit rates, and the latency requirements for each of the data products generated from the payload data. The model also receives a set of parameters containing data used by the contact efficiency model and by the latency simulator modules, which are explained in the following section. The final input parameters are the start and end times of the simulations.

Model Architecture

The constellation model architecture can be divided into two parts, one for the revisit and refresh and one for the latency, as described in Figure 2. Inputs are represented in yellow, modules in purple, and final outputs in green. The modules that fall within the Optimizer are grouped in the red box.

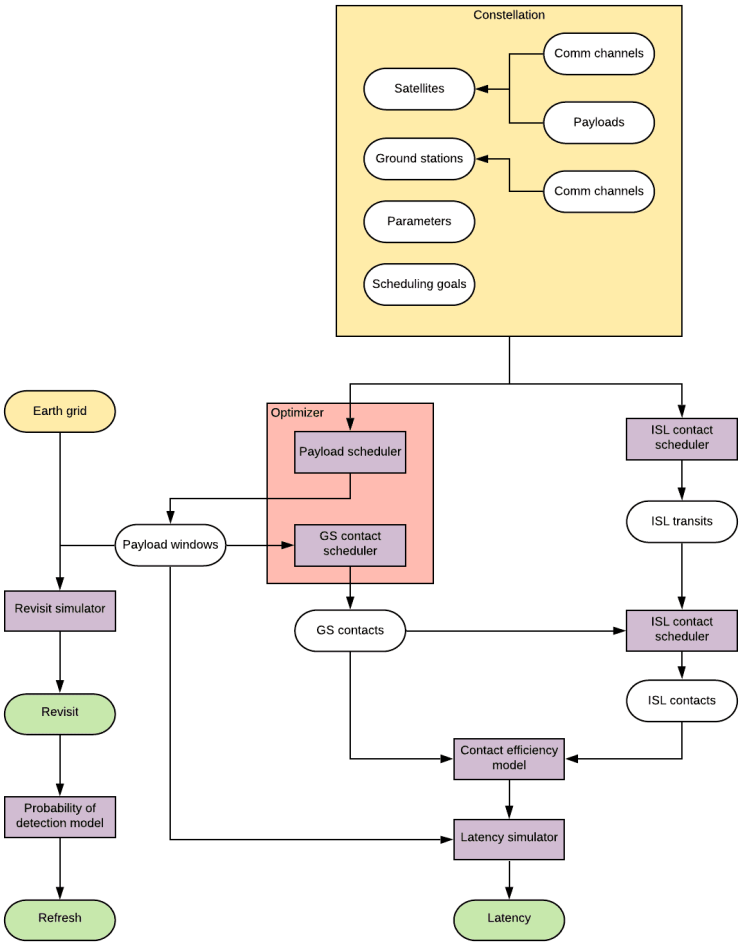

Figure 2: Constellation model architecture.

For revisit and refresh calculations, the user defines an Earth grid with the points at which the revisit rate will be calculated. The definition of this grid is dependent on the customer's use case and their area of interest. Then, the code makes an API call to the Optimizer to retrieve the payload schedule. The Optimizer is Spire's operational planning tool, and it is used both for operations and for simulation purposes. The next step in the process is the revisit simulator. For a given payload, with the payload schedule and the inputs (e.g., TLEs of the satellites, footprint shape of the payload, etc.), the revisit simulator generates the orbital path for each satellite and calculates the frequency of data collection for each point in the defined Earth grid. This collection frequency is used to calculate a variety of revisit statistics for the constellation (e.g., revisit heatmaps, revisit statistics per latitude, etc.). If desired, there is also an option to provide a probability of detection model to translate the revisit statistics into refresh statistics.

For latency calculations, the code makes two calls to the Optimizer, followed by one to the ISL scheduler. The first call to the Optimizer generates the payload schedule while the second call schedules the ground station (GS) contacts. For GS contact scheduling, the Optimizer first generates the transits (visibility period between a satellite and a ground station given the geometric and regulatory constraints) by processing the inputs and then schedules the best contacts among them. This will be presented in further detail in a later section of this paper.

Once the payload and GS contact schedules are complete, the ISL scheduler uses the geometric and regulatory constraints to generate ISL transits for the satellites. ISL contacts are then scheduled using the approach described in the Journal of Aerospace Information Systems.[1] The scheduled payload and contact operations are then fed into the latency simulator, which models how data travels through the constellation. The duration of the simulation is divided into time steps, and the simulation of each time step consists of (1) calculating the data generated by the payloads for each satellite and (2) modeling how that data is transmitted through the constellation.

To simulate how data is transmitted through the constellation, the latency simulator uses a contact efficiency model, which considers that anomalies can and will occur during a contact and that some contact time will not be useful for data transfer. This non-transfer time includes pointing of the ground station, contact acquisition, and communication of the current health of the satellite and subsystems. The size of the individual data packages, the data routing strategy, and the data package build time (i.e., the time needed for the satellite to collect and then bundle the data into files for transfer) can be modified in the input parameters for the simulator. The simulation of data transmission through the network is then used to compute several metrics (e.g., latency statistics and heatmaps, data volumes, etc.) for the constellation.

Use Case Example

A case with four ISL satellites is presented in this section to illustrate the outputs of the constellation model. The four satellites are located in different sun-synchronous orbit (SSO) planes with a right ascension of the ascending node (RAAN) increment of 30 degrees and with the same true anomaly. Their orbits are presented in Figure 3. Note that nine ground stations are used for this example with two payloads per satellite: PAYLOAD-1 and PAYLOAD-2. The payload windows in this example cannot be scheduled at the same time. Figures 4 and 5 present the payload schedule created by the Optimizer. The example uses a minimum operational time of 15 minutes and a duty cycle of 30% for PAYLOAD-1 and a maximum operational time of 30 minutes and no limits in the duty cycle for PAYLOAD-2.

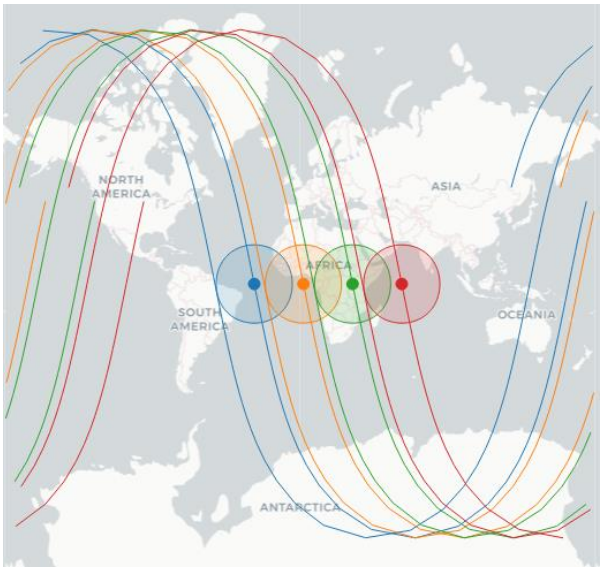

Figure 3: Orbits of the four satellites used for the example.

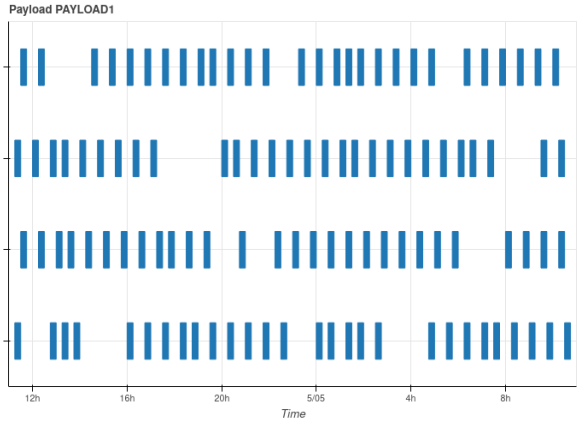

Figure 4: PAYLOAD-1 schedule.

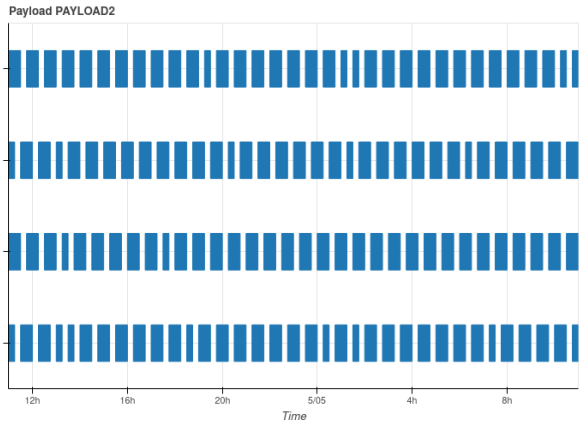

Figure 5: PAYLOAD-2 schedule.

Figure 6 illustrates the GS/ISL contact schedule created by the Optimizer for the example constellation. The GS contacts are represented in green, while the dark red lines represent a transmission ISL contact, and the rose lines represent a reception ISL contact.

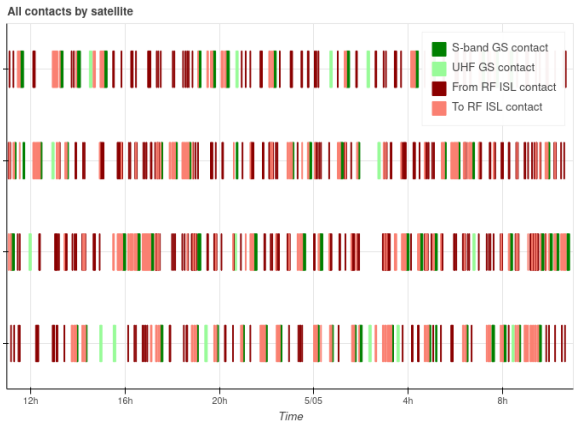

Figure 6: GS/ISL contact schedule.

When the simulation ends, the model provides the memory state (i.e., the amount of data of each data tier that the satellite stores) for all the satellites at each moment of the simulation. For this example, PAYLOAD-1 corresponds to tier 0, which has a higher priority for data downloading than PAYLOAD-2 (tier 1). The following Figure 7 shows the memory state of one of the satellites from the modeled constellation; data is generated at a different rate for each tier and is transmitted through green GS contacts and dark red ISL contacts and is received through rose ISL contacts.

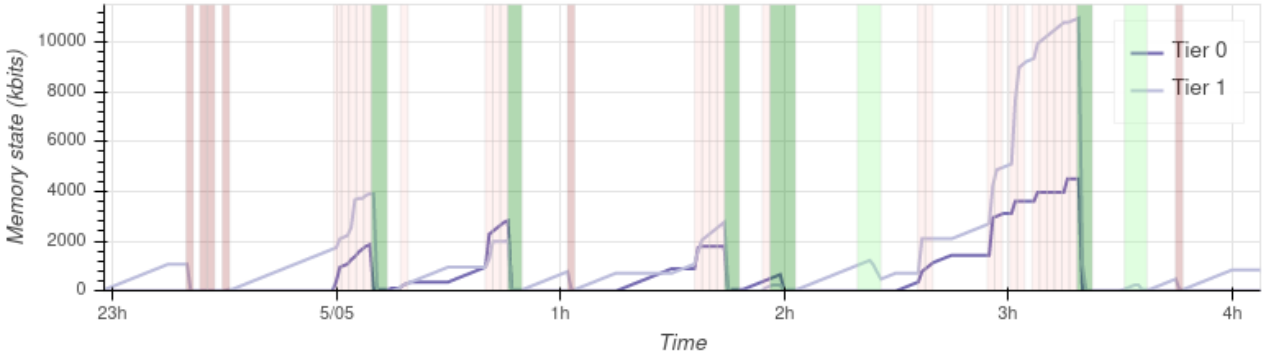

Figure 7: Memory state for a single satellite from the example.

The latency percentile curves shown in Figure 8 allow analysis of latency distribution at a glance. The latency percentile curves show in the y-axis the percentage of data that has a latency below the x-axis value, and each payload has an associated curve. In this example, PAYLOAD-1 has better latency because its data priority level is higher than that of PAYLOAD-2.

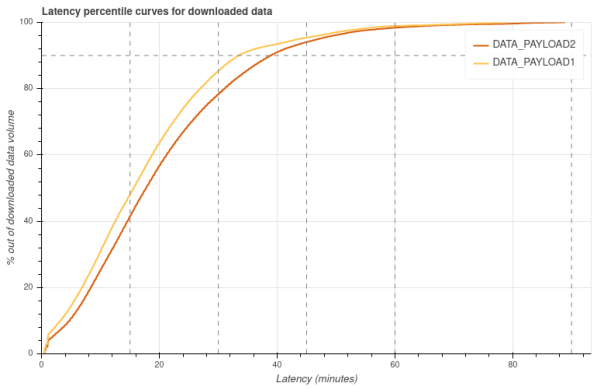

Figure 8: Latency percentile curves.

The constellation model is also able to provide the revisit and refresh metrics for the constellation. Figure 9 presents a revisit heatmap for PAYLOAD-2, and Figure 10 presents the revisit statistics per latitude for the same payload. A circular footprint with a 2000km radius is used for this simulation.

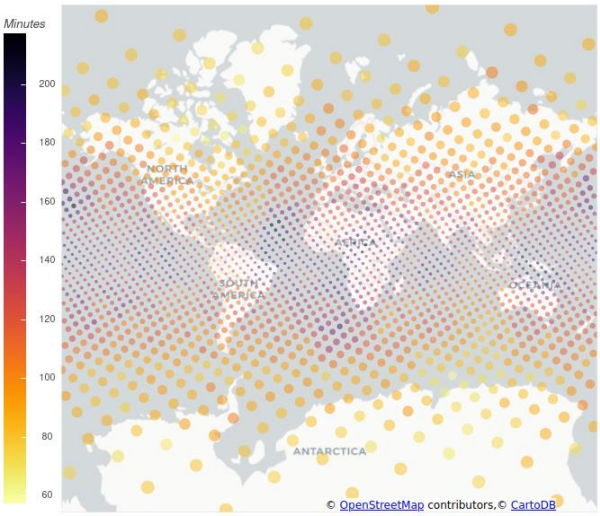

Figure 9: Average revisit heatmap.

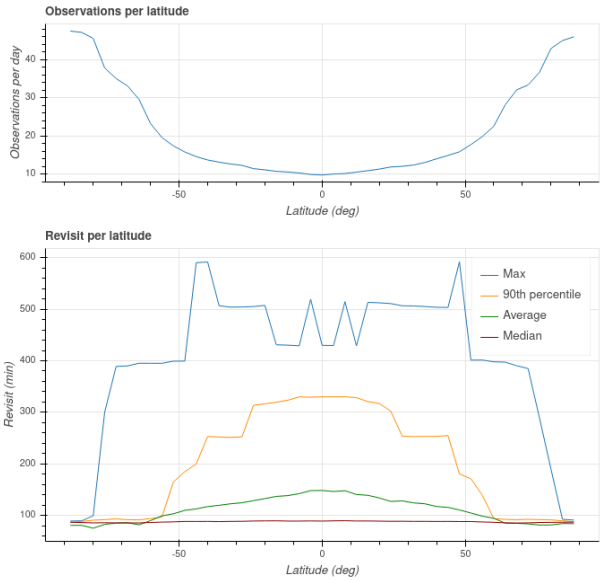

Figure 10: Revisit statistics by latitude.

The constellation model has been validated using real revisit and latency data from the current Spire constellation, and the average relative error between the predictions and reality is less than 10%.

CONSTELLATION OPERATIONAL PLANNING (THE OPTIMIZER)

The system that coordinates Spire constellation resources, known as the Optimizer, is responsible for maximizing the efficiency of the constellation and the value it produces through intelligent scheduling. The system consists of a set of automated schedulers that are capable of determining when constellation assets should perform operations such as GS contacts, ISL contacts, payload collections, and orbital maneuvers.

The schedulers operate on a batch processing model, allowing users to submit scheduling jobs to the system for processing and retrieve the results through the 'Optimizer Service API' upon completion. The system is separated into two queues: one for production scheduling and another for simulations. Production jobs generate the true constellation schedule and the Service API gives these jobs priority access to the solvers. Simulations, such as those performed in support of mission design, are run during any unused time which in practice is abundant. All jobs run asynchronously and their results are stored in a database for future analysis.

Optimally scheduling constellation operations is (at a minimum) an NP-Hard problem. The schedulers operate by encoding input problems as MIP problems which are solved using IBM's CPLEX solver. CPLEX

is fast and robust, meaning the models (which include hundreds of thousands to millions of variables) are solved to very tight optimality bounds, to within a few percentage points of the global optimal solution, in minutes. Wrapping the CPLEX core, the solvers are written using the Rust programming language and utilize FFI-bindings to the CPLEX C API for low-overhead interoperability.

Payload Scheduling

The 'Spire Payload Scheduler' is an advanced satellite constellation scheduling system which sees active production use. Within Spire, it is known as 'SPORE' (Scheduling Payloads for Objects Revolving Earth). The system performs multiple functions such as payload deconfliction, area-of-interest targeting, and general power management (through flexible duty ratio constraints) in order to schedule optimized data collection for Spire's entire constellation of satellites. SPORE is fully automated and incorporates information on each individual satellite's power, coverage, and conflicting hardware subsystems to balance collections across time. This works not only for a single satellite, but also across the Spire fleet as a whole, ensuring that redundant collections are minimized while novel collections and revisit targets are achieved.

Figure 11 shows payload operations when the constellation is instructed to deprioritize data collection over the Atlantic Ocean. The satellites instead use this flight time to preserve power in order to provide better coverage over the rest of the globe. Automatic and coordinated scheduling makes prioritization like this possible with declarative, user-provided configurations.

Figure 11: Payload schedules designed to avoid collections over the Atlantic Ocean.

Contact Scheduling

The results of the payload scheduling optimization are then fed into a pipeline which generates the set of candidate ground stations accesses, as illustrated in Figure 12. The Spire contact scheduler selects the optimal set of accesses to ensure that all payload data is downlinked quickly and efficiently. The system is capable of scheduling the entire constellation in less than five minutes, is orbit-aware, and accounts for constraints such as RF licensing, compatible frequency bands, data transfer directionality, and link protocol when determining viable accesses.

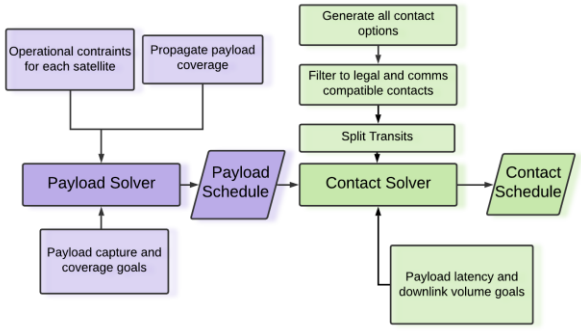

Figure 12: Constellation model format incorporating the payload scheduler (SPORE) and the contact scheduler.

Incorporating the payload schedule into the contact scheduling process allows Spire to allocate contact time to assets in the fleet based on the volume and value of the data collected. The added context provided by the payload schedule enables the scheduler to optimize for business objectives such as decreased latency between data collection and data downlink. Satellites generating data which is highly sensitive to latency may be given a large number of small contacts by the contact solver, while high-value but latency-insensitive data is scheduled for transfer during less contentious accesses to ground stations.

Given the heterogeneous nature of the Spire constellation, it is important to have an interface mechanism for the schedulers that is flexible enough to accommodate a wide range of possible constellation architectures. The Spire schedulers are designed to be 'intent driven,' operating on declaratively defined models that describe the 'shape' of the constellation (i.e., its ground stations, satellites, and payloads) and the performance goals. The models do not receive input parameters on how to achieve these goals; the schedulers are responsible for mechanizing the users' goals and finding optimal plans of action to satisfy them.

Model Design

All schedulers work with a single model format, known as the 'Unified Model,' that encapsulates all of the information required to represent the constellation at a given point in time. Schedulers return results in this format as well, describing the constellation state after an optimal plan of action has been calculated, as illustrated in Figure 13. By standardizing the design of the input and output formats across Spire's solvers, users are empowered to compose the solvers to best meet their scheduling needs. This standard format also enables interesting workflows that would not otherwise be possible, including very long-duration simulations which cannot be solved monolithically, but can be handled as a sequence of smaller problems.

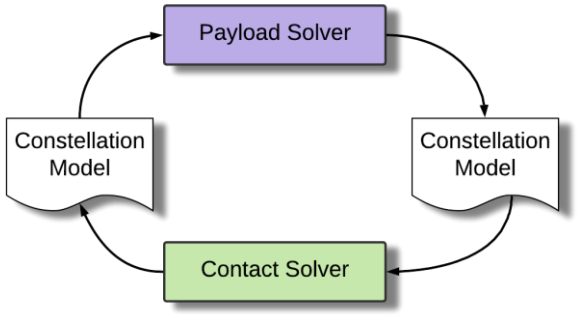

Figure 13: The Unified Model leverages a standardized input and output format.

What, when, and how assets can communicate is a key operational concern for the constellation. The Unified Model is able to represent these constraints using the communications configuration model, which is a standardized format for describing possible links between assets, including licensing and performance data. These channels allow Spire to model the communications of each asset at a very fine granularity. Configurations include not only the frequencies which are enabled for each asset and the directions in which they transfer data, but also regulatory and geometric constraints that must be taken into consideration in order to perform compliant links between assets. Beyond simple static licensing rules, the communications configurations are also able to capture more advanced constraints such as minimum separations between orbiting objects, which enables Spire to respond to coordination requirements from third parties.

The schedule for the production constellation is populated by a set of periodically executing tasks. Each task will populate the schedule for one constellation operation type, such as ground station contacts or payload activity. As seen in Figure 14, tasks run as a pipeline that integrates with the required Spire systems and performs four major functions: extracting constellation data into a scheduling problem, solving the problem via the Service API, publishing the result, and, in the case of failure, retrieving and publishing a fallback solution.

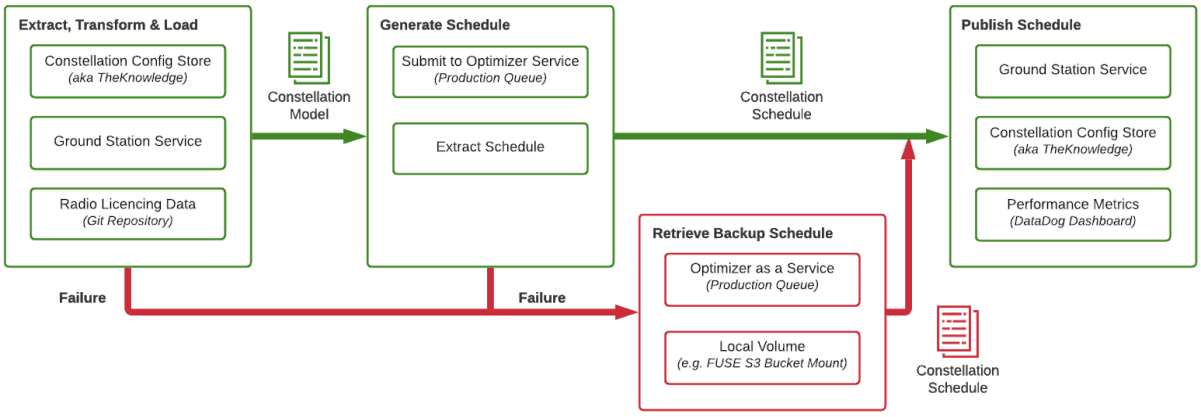

Figure 14: Task flow for the scheduling pipeline.

Each of these stages is able to run independently of the others, though they are run as a single sequence of operations by default. Scheduling is entirely automatic with the pipeline on AWS Fargate tasks triggered using periodic CloudWatch events. The scheduling pipeline also generates failover solutions with the publication step disabled and an offset planning period. Having a precomputed solution available provides robustness in the case of solver failures. As part of the final publication step, the Spire monitoring service produces comprehensive metrics detailing solution quality and key health data.

As Spire expands the constellation and continues to add capabilities, the scheduling systems will evolve. One future addition to Spire's scheduling service is the ISL solver. Adding this capability emphasizes the flexibility and adaptability of the Optimizer Service. The ISL solver will be an extension of the standardized constellation model format and process, and as such the scheduling systems will be able to accommodate new problems by simply adding the ISL-specific solver onto the current pipeline. Adding a new solver to production involves adding a Service API endpoint to call the solver, updating the constellation model to include the new solver channels and plan contacts, updating the pipeline to extract and publish the contacts, and adding the production-ready solver implementation to the workers. Because of the modular design of the service however, there is no need to build or provision new infrastructure to handle the additional scheduling requirements.

PERFORMANCE MONITORING

Spire's monitoring application tracks key performance and correctness metrics for the systems, including bi-directional contact time, total contact time, the number of payload windows scheduled, and solve time. A high-level summary dashboard focused on the cloud computing system runs continuously, detailing both infrastructure usage (i.e., CPU load, memory usage, network traffic, etc.) and scheduling performance. It provides automated alerting integrated with a paging and on-call escalation system. In the event of a service issue, alerts providing situational awareness are dispatched through Spire's internal messaging application, and if metrics were to fall into identified critical regions, system pages route the alerts directly to on-call engineers to allow for immediate response to a potential problem.

For many of the targeted metrics, simple thresholds work well and are reliable for monitoring purposes. However, the monitoring system also supports more advanced application monitoring when required, including outlier analysis, trendline prediction, and liveness checks. All of Spire's alerting is configured declaratively, which makes keeping alerts up-to-date and synchronized across deployments of the service simple and ensures that any new features are pre-equipped with the appropriate monitoring framework. Alerts are passed through to Spire's Operations Center for viewing and analysis by a team of operators. Figure 15 shows an example Spire Operations Center dashboard for a single satellite, providing a visual representation of the health and operational status of the spacecraft.

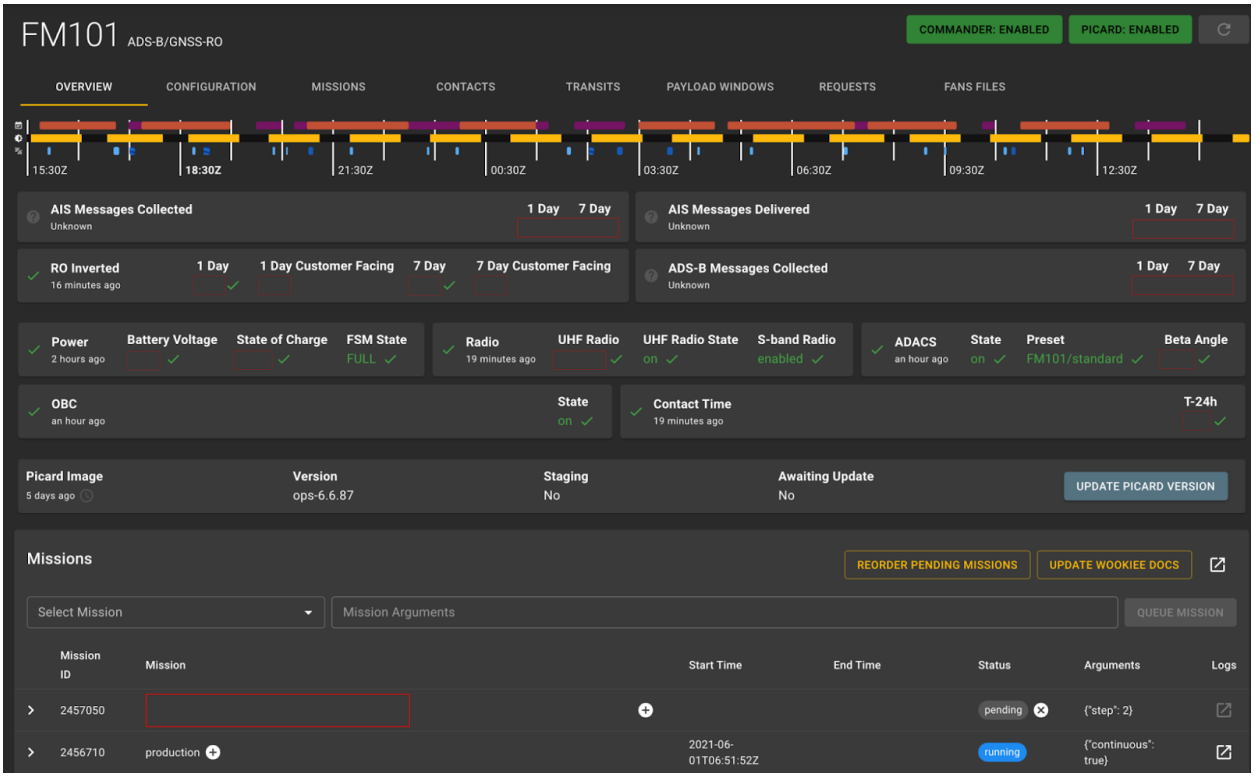

Figure 15: Example Spire Operations Center page for a single satellite.

ENABLING CUSTOMERS WITH STANDARDIZED USER SERVICES

Spire has created a resilient and fully integrated satellite, ground station, and cloud-based operations platform to enable efficient data collection from space. Spire's Space Services extend the capabilities of this platform to a wide range of customers via comprehensive user services, enabling users to easily integrate with the constellation management systems and tools for scheduling payload operations, interfacing with the satellite bus, uploading and retrieving payload data, and providing execution environments for customer software hosted in space. These user services include a suite of APIs, libraries, and operating systems that enable seamless integration and operation of customer payloads and software within the Spire constellation.

Spire Space Services consist of three primary offerings as outlined below. The nature of the offerings is relevant as it dictates the scope of the user services provided which are referenced herein.

1. *Software in Space:* customer-deployed software on existing satellites, leveraging Spire software defined radios (SDRs) and payload systems to test and scale applications without the need to launch a dedicated spacecraft.

2. *Payload in Space:* customer payloads hosted on the Spire Low Earth Multi-Use Receiver (LEMUR) satellite bus, leveraging Spire's end-to-end launch services and operational tools to rapidly deploy, demonstrate, and scale systems into production.

3. *Solution in Space:* purpose-built end-to-end solutions for customers looking to partner with Spire, leveraging heritage design architecture and in-house development and manufacturing expertise.

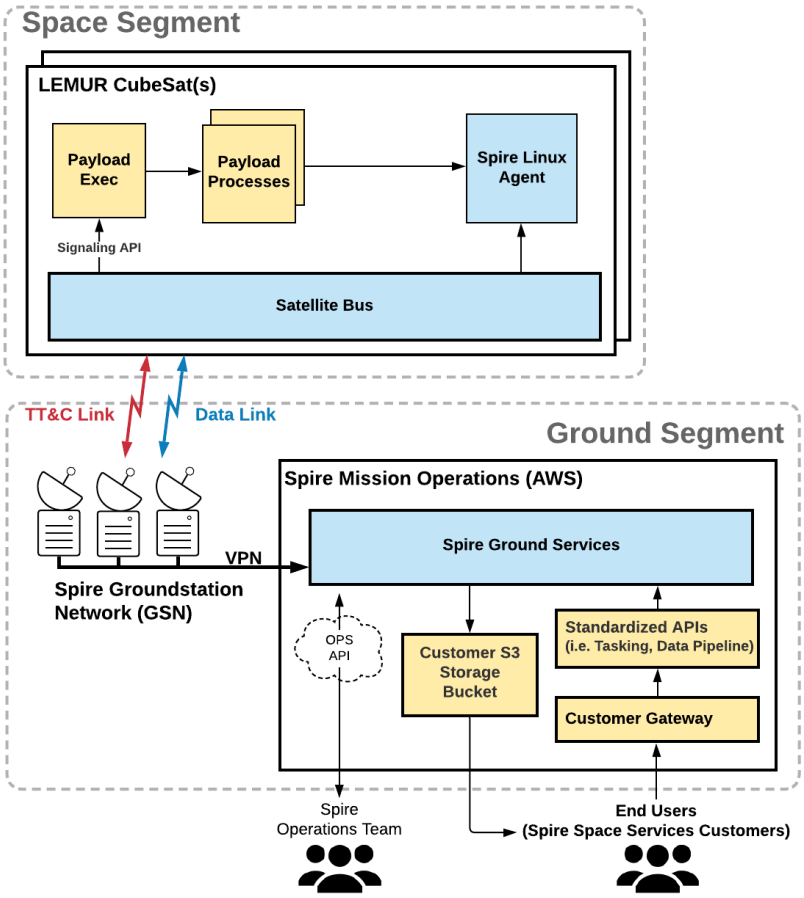

Figure 16: Spire User Services

Scheduling API

The ‘Scheduling API’ provides access to the payload scheduler, SPORE, in the Optimizer system. This allows customers to generate candidate payload windows automatically in the same way that Spire schedules the entire constellation. The windows generated by this service can be submitted to the Tasking API for execution on orbit.

Tasking API

The Spire constellation is controlled via a globally synchronized calendar of fixed-duration operational windows. Each window has a fixed start time, a fixed duration, a type, and additional window-specific parameters that tune its operation. The master schedule is stored in Spire-controlled terrestrial infrastructure. Relevant windows are synchronized to each satellite during regular maintenance procedures and executed by each satellite's onboard controller.

The ‘Tasking API’ is one of several standard RESTful API web-services Spire offers, providing customers a set of endpoints used to task, configure, and manage payload operations. Users interact with the Tasking API to create ‘payload windows’ which define the start and end times of an operation for a given window type (payload operation type), as well as set the configuration or parameters of the desired operation. The API can be used directly for manual scheduling of individual windows or, more commonly, as a service for automated system scheduling.

A single payload window may consist of multiple steps that are scheduled together. For example, a standardized software defined radio payload task (e.g., ‘PAYLOAD_SDR’) can orient the satellite, record a data sample, and perform an analysis of the recorded sample file in one schedulable unit. Additionally, the Tasking API can be used to upload software or any other arbitrary file to the user's payload.

Spire Linux Agent

The ‘Spire Linux Agent’ is a daemon that Payload in Space customers install and run on their payload to enable seamless integration with the Spire LEMUR satellite bus.

The agent binaries (for supported architectures) and source code are provided to Payload in Space customers prior to launch to support their development. To interface with the Spire Linux Agent, Spire provides a C software development kit (SDK) and a Python SDK. For other programming languages, users can make HTTP requests directly to the agent. The daemon provides access to the Data Pipeline API for Payload in Space customers to manage data to and from the payload.

Data Pipeline API

The ‘Data Pipeline API’ allows Payload in Space users to download data from their payload to their ground-based data storage in AWS S3. This API was designed to abstract the complications of managing a disruption-tolerant network from the end user and provide a simple, always available way to access the data pipeline.

The Data Pipeline API is made available by the Spire Linux Agent and associated SDKs (see Spire Linux Agent section above).

Signaling API

The ‘Signaling API’ provides payloads hosted on the LEMUR satellite bus the ability to receive and act on events generated by the bus, such as the start of a payload window. The Signaling API currently supports Linux-based payloads with an SSH daemon running. The satellite bus will execute an executable developed by the customer (i.e., ‘payload_exec’), with the configuration specified for a given event.

The Signaling API defines the interface that payloads must expose for the satellite bus to inform the payloads of upcoming events. The API consists of:

1. A payload executable used to respond to satellite bus events. This executable will be called by the bus using SSH.

2. Conventions of where payload window configuration files and uplinked packages are placed on the payload file system by the satellite bus.

3. Window configuration file schemas provided by the satellite bus for payload executables to use. These window configuration files contain bus information about the signal as well as relevant data passed through from the Tasking API.

4. Argument schemas that payload executables must accept to handle satellite bus signals.

Transmission of data from the payload to the ground is handled via the Data Pipeline API described above. It is the responsibility of the customer payload to implement an executable that can be called with certain parameters after the window configuration file is placed on the

payload file system. This executable will interpret the window configuration file and perform the necessary payload actions.

CONCLUSIONS

In the months and years to come, Spire anticipates that the constellation will continue to diversify with more ground stations and satellites with unique payloads, capabilities, requirements, and constraints. The constellation modeling, monitoring, and operational management systems outlined in this paper equip Spire and Spire's customers with a flexible set of tools that are adaptable and scalable for future generations of satellites and constellation configurations. With such a wide number of variables and limitless trade-offs, Spire strives for continuous improvement to further optimize the efficiency, flexibility, and production capabilities of the network. Spire recognizes that the data collected and the products produced varies by payload, data type, use case, and user. Exposing these tools to customers through standardized APIs and continuing to improve access, adaptability, and responsiveness of these systems will enable seamless and scalable deployment of next-generation payloads and capabilities.

ACKNOWLEDGMENTS

Spire acknowledges that some of the tools and technologies outlined in this paper were developed with funding from the United Kingdom Space Agency (UKSA) through the European Space Agency (ESA) ARTES Pioneer program.

Spire also acknowledges all the employees that contributed to the development and sustainment of the tools and technologies outlined in this paper.

REFERENCES

1. Lowe, C.J. and M. Macdonald, “Resource-Considerate Data Routing Through Satellite Networks,” Journal of Aerospace Information Systems, vol. 14, No. 8, August 2017.