Adaptive Autonomous Communications Routing Optimizer for Network Efficiency Management

Robert Richards
Stottler Henke Associates, Inc.
951 Mariners Island Blvd., Suite 360
San Mateo, CA 94404
Richards@StottlerHenke.com

Jeremy Ludwig
Stottler Henke Associates, Inc.
951 Mariners Island Blvd., Suite 360
San Mateo, CA 94404
Ludwig@StottlerHenke.com

Abstract— **Maximizing network efficiency for NASA's Space Networking resources is a large, complex, distributed problem, requiring substantial collaboration. These networking resources include the Deep Space Network (DSN), Space Network (SN), TDRSS spacecraft, Near Earth Network (NEN), and future Exploration Destination networks, which are being integrated under the NASA Space Communications and Navigation Program (SCaN). This paper deals primarily with adaptive autonomous network management to schedule communication events between space and space/ground assets. The three central problems of interest in scheduling space communications are: 1) constraints defining missions' communication needs are complex and often come in competing shades of gray; 2) resources are heterogeneous, expensive, and frequently oversubscribed, resulting in conflicted schedules; and 3) mission criteria for a "good schedule" that meets their objectives vary widely from mission to mission, making it difficult to satisfy mission preferences. In this paper we describe the development of a prototype networking and scheduling framework that facilitates the development of more intelligent and optimizing scheduling algorithms within a mixed-initiative architecture. This framework is capable of representing the complex and diverse constraints found in the space communications scheduling domain, allowing for a valid schedule to be created. Once a valid schedule has been produced, a multi-objective resource optimizer refines the schedule to maximize mission satisfaction. Mixed-initiative conflict resolution helps to address any remaining scheduling issues. An example combining scheduling for the NEN and TDRSS is used to illustrate the features of the existing prototype and outline future work.**

TABLE OF CONTENTS

1. INTRODUCTION

Increasing NASA's Space and Ground networking efficiency is a large, complex, distributed challenge. These networking resources include the Deep Space Network (DSN), Space Network (SN), TDRSS spacecraft, Near Earth Network (NEN), and future Exploration Destination networks, all of which will be integrated under the NASA Space Communications and Navigation Program (SCaN). The technologies to increase efficiency must be adaptable to a variety of network operating environments, ranging from the long latency limited bandwidth of deep Space Communications to near Earth environments with traffic flow over global partner assets, and the future internet; that is, the entire space network that includes ground to ground segments, in addition to space-ground links. In addition, to provide a solution that will be adopted by the now mostly-separate DSN, NEN & SN communities, the solution must be flexible in its development and deployment so as to leverage as much commonality across the communities as possible and allow for maximal re-use of what is already working.

Stottler Henke's solution, termed AaCRONEM (Adaptive Autonomous Communications Routing Optimizer for Network Efficiency Management) provides the capabilities and flexibility required. For example, our technological foundation dealing with adaptive autonomous network management can be leveraged as needed, while our current user interface is designed to be adapted easily, so that it can be modified to mimic current user interfaces if desired. Almost all Stottler Henke solutions interface with other systems, so AaCRONEM will, similarly, interface with current scheduling tools and/or data sources as needed. The AaCRONEM prototype, and our related Air Force Satellite Control Network (AFSCN) scheduling solution also under development, already demonstrate sophisticated, unique and innovated capabilities not otherwise available.

Stottler Henke's Aurora scheduling technology, which is already in operational use by NASA, provides much of the core functionality for AaCRONEM. AaCRONEM also utilizes as a component OPNET Technologies' network simulation tool, *Joint Communication Simulation System* (JCSS), in order to create optimal schedules for network resources based on planned utilization, and then to adapt based on how the actual utilization differs from the plan in real-time, as well as adapting to network anomalies.

There are three central problems in scheduling space communications: 1) Constraints defining missions' communication needs are complex and often come in competing shades of gray; 2) resources are heterogeneous, expensive, and frequently oversubscribed, resulting in

978-1-4577-0557-1/12/$26.00 ©2012 IEEE

conflicted schedules; and 3) mission criteria for a "good schedule" that meets their objectives vary widely from mission to mission, making it difficult to satisfy mission preferences.

These problems persist in addition to the non-trivial challenges in maximizing the efficiency of ground-to-ground segments of the entire network. Through combination of our technology with OPNET JCSS, our solution incorporates the entire space networking domain, while leveraging many of the state-of-the-art solutions for maximizing network efficiency already existing in JCSS. This allows us to simulate and test our enhancements in a global sense. JCSS is a DISA distributed simulation tool, built on the COTS OPNET Modeler platform. OPNET Modeler provides best-in-class network modeling and simulation capabilities including extensive support for analysis of wired, wireless, and satellite communication technologies. As an industry standard, Modeler/JCSS allows us to develop a COTS/GOTS tool that it is logical for those working on other promising techniques to integrate with, thus providing a framework for incorporating the best techniques for a better, more complete solution.

These three central problems in scheduling Space Communications are considered in more detail below.

Space Communications Constraints

The communication requirements that vary from one mission to another, combined with the range of resource capabilities, result in an array of complex request constraints. A mission may require, prefer, or allow:

- N out of M requests per day at ground station G (e.g. AURA requires 2-3 contacts per day at Wallops out of the ~14 per day that it requires overall on the NEN).
- Not to schedule the same aperture on consecutive orbits (e.g. AQUA on the NEN).
- A mixture of services required across a given day, where the request order does not directly matter but the service usages may have their own separation requirements (e.g. EO-1 requires at least 3 S-Band supports between 12:00 and 21:00 out of the 6 or 7 per day, where S-Band contacts need to be at least 5 hours apart but at most 8 hours apart on the NEN).
- Requests to share antennas if the same antenna orientation can hold both craft in view (e.g., Mars communications). In such cases they can share a downlink, but not an uplink (MSPA protocol for the DSN).
- Requests to be handed off from one ground station to another in order to meet the duration requirements; some handoffs require a certain degree of overlap to complete successfully (segmentation protocol for the DSN).
- Some requests actually need to be handled by two different stations in order to perform a triangulation (DDOR protocol for the DSN).

This is a sample of just a few of the disparate constraints that need to be considered by an adaptive network management system. This basic complexity is compounded by the fact that these constraints are a requirement for some missions, a preference for others, and permissible for conflict resolution for still others. Furthermore, this is a dynamic environment: both the communication demands and network supply change over time, and they may deviate significantly from the original expectations when planning occurred.

The other challenge is that, as varied as the known constraints are, it is likely that more constraints will be required in the future as new types of resources are added, and new mission needs are identified.

Conflicted Schedules

Communication resources are frequently over-subscribed. This is partly because they are limited in number, partially because they are not homogeneous, and in part because they cannot readily be added as demand goes up. The ever-increasing number of missions virtually guarantees that this will remain an ongoing problem.

This problem is aggravated by two factors: communications can only take place in specific view-periods, when an antenna is in view (e.g. this is especially problematic for low earth orbit satellites on the NEN network), and furthermore, missions tend to request more passes than they technically need. These dynamics both increase the likelihood of conflicts, the first by adding constraints to the scheduling problem, and the second by adding passes that are desirable but not necessary.

Conflict resolution is, frequently, a highly manual process in some of the systems still in use. This is partly due to current software limitations, and, in part, because many of the constraints cannot actually be modeled in current systems.

Mission Satisfaction

Different missions have different criteria for whether they are satisfied by their pass allocations. In some cases maximal communication duration is preferred. In others, they may care more about the quality of the communication time allocated – whether the allocations are at antennas with the highest possible PCA elevation angle, are at a given time of day, and/or are supported by a particular service. Of course, this "quality" definition varies as well.

This can make it difficult for the adaptive network management system to effectively balance the requirements and the complex constraints, both hard and soft, and for it to

allocate limited resources in such a way as to make the various missions as happy as possible with the end result.

Conclusion

Currently much of this complexity is handled by the human schedulers, who use their own knowledge to ensure: that the more obscure constraints are correctly satisfied; that the appropriate tradeoffs are made in solving conflicts; and that the missions' most critical needs are met. However, this is not a practical long-term solution as the problem complexity increases.

There are two reasonable long-term solutions to this problem: to automate as much of the scheduling process as possible, and to allow the missions to do more of their own scheduling / conflict resolution. The advantage of the first approach is that computers handle scalability much better than do people. The advantage of the second is that it shifts much of the burden of requirement definition and fine-tuning back onto the missions, which as-is have the best understanding of their requirements and preferences. According to DSN's "Scheduling SW Requirements", DSN is already pursuing this goal.

Success for the former approach would require correctly modeling the complex constraints that dictate the request relationships, and explicitly considering mission needs. Success for both approaches would require significant conflict resolution support.

All of these factors highlight the need for a sophisticated scheduling system: capable of capturing the constraint complexity required to model the interactions across different passes and complex resources; that can assist the user throughout the scheduling and conflict resolution process, both in terms of offering options and advice, and in terms of insuring that the different mission needs are being met and balanced; that can optimize the schedule while balancing tradeoffs among these various constraints and their relative importance for different missions.

AaCRONEM should *not* be seen as a solution that is trying to subsume all the current scheduling systems that are utilized by the separate communities, but as a framework that can be leveraged to complement what is already available. For example, if the DSN community has specific scheduling requirements that are beyond what is in Aurora, but are already implemented, AaCRONEM could call that system, appropriately. Thus AaCRONEM will provide integration between the various systems to create a result, which leverages the best of all worlds to create a superior integrated common solution.

Figure 1 below shows Aurora's more standard default interface on the first and the Air Force interface as implemented by Stottler Henke second. Note that the interface is decoupled from the underlying scheduling engine, so either interface could be used to see the same results.

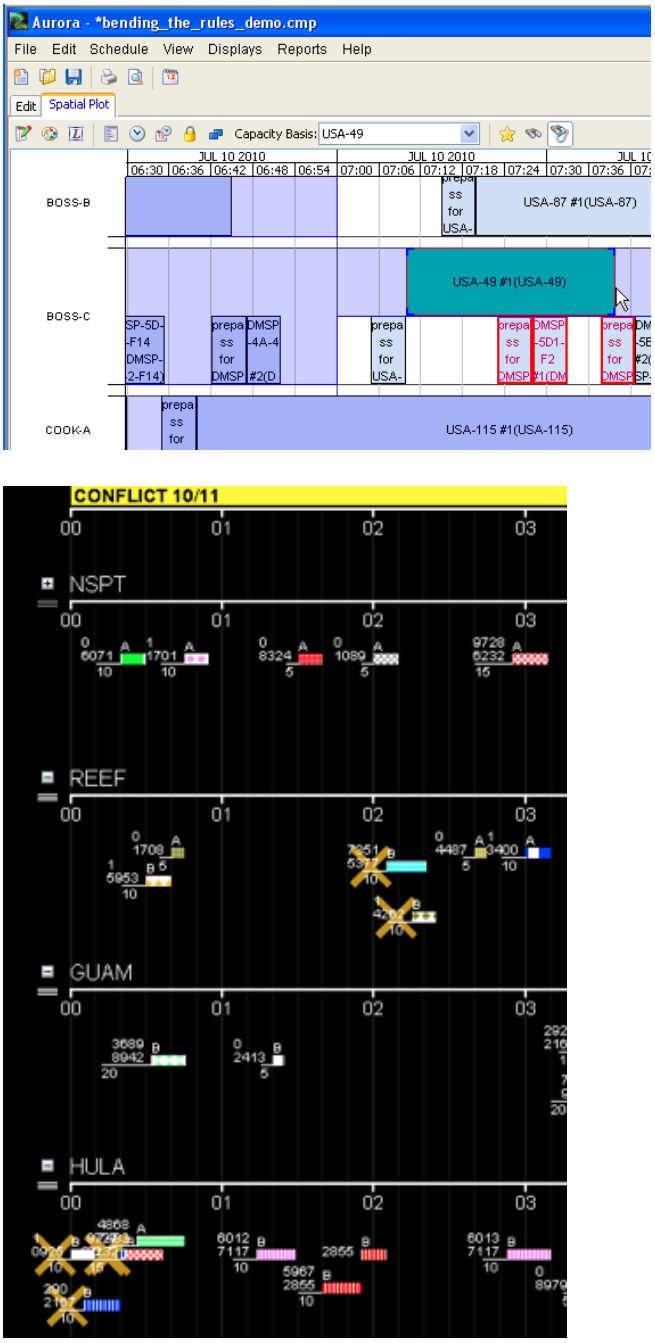


Figure 1: Example of Adaptability of the User Interface

2. System Description

Figure 2 shows the high-level architecture of the proposed system, including user interactions; note that the dashed arrow indicates the input into the system during actual execution, where AaCRONEM will compare actual performance against predicted performance and determine if any adjustments are needed. Each of these components is described in greater detail below. Note that the scheduling module may consist of multiple underlying schedulers, some of which could be external; e.g., in certain DSN

situations the current DSN scheduling engine could be called as one of the schedules.

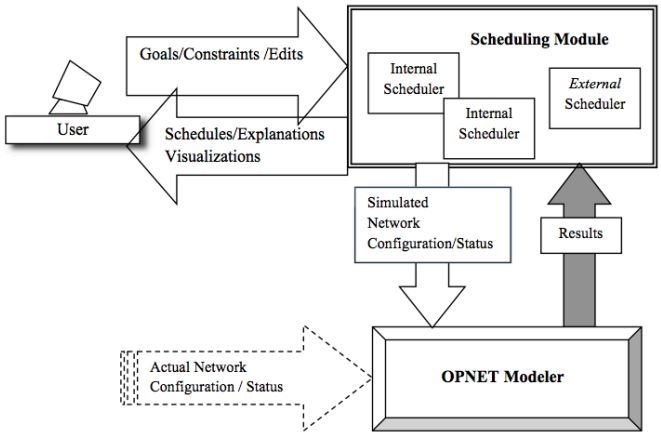


Figure 2: High-level architecture and user interaction

More details of the overall Scheduling Module are shown in Figure 3. This module is assembled leveraging our existing intelligent scheduling system architecture, Aurora (http://www.stottlerhenke.com/products/aurora), which provides for customization of each decision point in the scheduling processing. For example, it already includes the notion of scheduling cycles, an evolving schedule, the need for a separate preprocessing module, handles resource usage profiles and visibility requirements, and provides for both pluggable resource/time window selection methods and satisfaction of temporal, spatial, and arbitrary constraints. Note that the box labeled "Different Algorithm Scheduler" might represent an NASA scheduler, such as the current scheduler used by the SN.

If there are multiple schedulers, then a preprocessor is needed. The roles of the *Preprocessor* are to 1) call each applicable scheduler, 2) grade each returned schedule, and 3) select the best schedule. Emergency service requests requiring attention immediately would be passed to the *Emergency Response Scheduler* for immediate scheduling, so that the appropriate tasks can be passed back and executed immediately. Otherwise, or for the remaining requests, the *Preprocessor* selects one or more appropriate schedulers. (These schedulers will all be independent, and therefore would easily fit into a distributed architecture with each scheduler having its own computing resources, if required.)

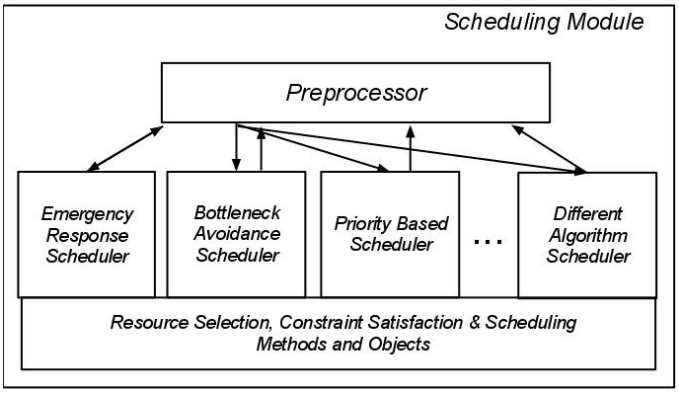


Figure 3: High Level Scheduler Architecture.

Figure 4 shows more details of the components of a Scheduling Module and their interactions.

Even with multiple internal schedulers, there will be sharing of scheduling components. In addition to the scheduling system core there will be several supplemental components that will be used in the primary scheduler and possibly other schedulers.

- *Extensible Constraint Model*, which will encapsulate the complexity of the diverse constraints.
- *Schedule Modification Rule Base*, which will encapsulate knowledge about how requests can be shifted and manipulated, and how to put them back.
- *Schedule Modification Case Base*, which will keep track of information about what types of actions are preferred in what scenarios, and update this information through time.
- *Mixed-Initiative Conflict Resolution Assistant*, which will analyze conflicted areas of the schedule to find possible solutions and then work with the user to solve the problem.
- *Multi-Objective Resource Optimizer*, which will perform local optimization on the schedule in an effort to improve mission satisfaction.
- *Execution Monitoring and Repair module*, which will monitor the actual overall space network under the auspices of SCaN to determine if any modifications are necessary to the plan due to the network state being different than assumed during the planning phase.

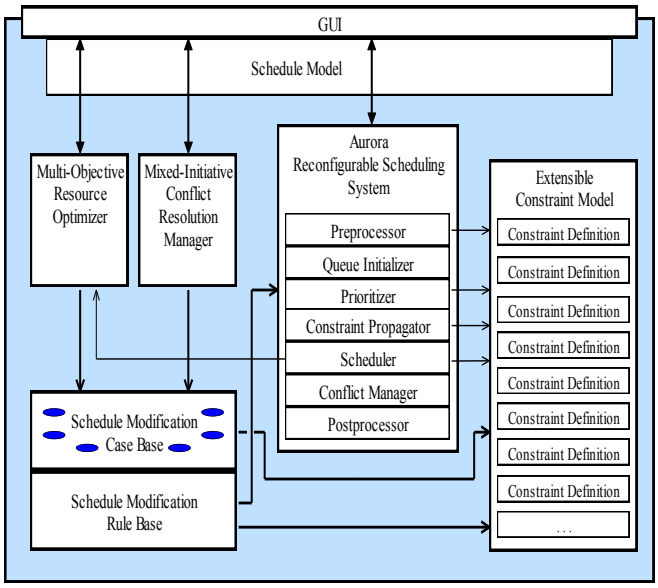


Figure 4: Detailed architecture for a scheduler

Extensible Constraint Model

There are two challenges involved in modeling the complex constraints that define the preferences and requirements of mission communications. First, a variety of known constraints impact the relation of timing across requests, resource assignments across requests, and which assignments should be preferred in what situations. Second, because the existing constraints are derived from the interaction between complex mission requirements and complex resource attributes, it is inevitable that new types of constraints will have to be accommodated in the future.

To encapsulate this complexity in a way that can readily be accessed from the appropriate points in the scheduling process, while still allowing for easy extensibility, we have a separate module that manages the complex constraints. This module will be expanded to handle constraints across sets of requests, and will know how to maintain the constraints, as well as what operations to take to soften them.

The extensible constraint model will encapsulate all known inter-request constraints. These constraints will not necessarily be between two specific requests (as in normal precedence constraints); some might link a number of requests (e.g., out of M linked events, N events need to be at ground station G within time period T). The alternative to this would be to break the constraints into more atomic versions, but this would involve maintaining parallel plans and then picking one (e.g., for the N events at ground station G, it could be broken down to the first and second event at ground station G . . . the first and third event at ground station G . . . the first and fourth event at ground station G . . .), but this would result in an exponential number of options complicating an already-challenging scheduling problem.

In the case of most of these complex constraints, it is far more efficient to model them in their complexity, and use sophisticated heuristics to satisfy them (e.g., in the N events at ground station G, the system might keep track of how many more events in the linked group had yet to be scheduled, and try harder and harder to schedule an event at G the fewer there are relative to the number required).

Schedule Modification Rule Base

The schedule modification rule base *is* a library of schedule manipulation techniques for use in conflict resolution and schedule optimization. The AaCRONEM system will enhance the schedule modification rule base already provided in the AaCRONEM prototype and the Air Force Satellite Control Network scheduler code base. It contains logic for common atomic operations such as: shifting a request earlier or later in the current view period; shifting a request to an earlier or later view period; and shifting a request to a specific resource or set of resources. All of these operations can be performed while paying attention to other requests (trying to satisfy constraints and not over-allocate resources), or not (usually as a first step in a multi-step modification).

The rule base is be able to combine these atomic operations into multi-step modifications such as shifting a series of spaced requests earlier or later; performing an N-way swap among different requests; and rescheduling a constrained set of requests starting with the bottleneck request and working out from there.

The rule base is also responsible for recording the state of each request before any attempted modifications, so that a request can be restored to its previous state, if necessary. This is especially important for exploring different options to present to the user without impacting the final schedule.

Schedule Modification Case Base

The schedule modification case base keeps track of past modification strategies that were either performed manually by the user or approved by the user. These are cross-referenced by details of the situation, most notably the user, constraints, and mission(s) involved. This helps capture both explicit and implicit information about the relative priorities different missions give to their different constraints.

For example, it might be perfectly acceptable for one mission to have a request scheduled fifteen minutes later than the maximum time indicated by the maximum separation constraints – but for another mission, such a decision might mean the loss of valuable data. Another example tradeoff would be the choice between getting 10 contacts on a given day, or 9 contacts with all preferences met.

What aspects of a request that missions care about vary broadly, and this case base helps capture such knowledge through time to make future conflict resolution and

optimization easier and more robust. AaCRONEM will leverage the case base functionality available from the AFSCN Scheduler code base, and depending on the priorities of NASA, develop a SCaN-specific case base.

Mixed-Initiative Conflict Resolution Assistant

The mixed-initiative conflict resolution assistant analyzes a conflict to find possible ways of solving it. It may either run in response to the user (e.g., the user selects a conflicted request and asks for resolution options), or function automatically. Normally, once the system is done producing a schedule, it will use down time as available to analyze conflict resolution options for any conflicts, so that it can be ready for a user request.

First, the conflict resolution assistant uses the schedule modification rule base to record the start state of the problem element. It will then look at the constraints involved and use the schedule modification case base to determine the range of possible options for rescheduling the problem request. Using the schedule modification rule base, it will try each option, recording any that worked. It then considers each view period where the problem request could potentially schedule while satisfying all of its constraints, and performs the same basic analysis process on the requests that are already scheduled at those locations (trying to move them out of the way by softening their constraints). Finally, if there are insufficient conflict-resolution options that succeeded in finding a solution, it will perform the second step again, but this time, instead of only looking at the requests in view periods that could satisfy all of the constraints, it will also consider requests in view periods that could satisfy most but not all of the problem request's constraints.

These steps give the assistant a set of potentially acceptable options. It will rank these options according to the case base preferences and the actual request parameters, and display them to the user on request. When the user selects the desired strategy, it will redo the requested operation, and present the user with the result. The assistant may also have a mode wherein the user can automatically resolve conflicts in the mission(s) over which they have full authority, based on softening criteria in the case base. This allows the system to fix any conflicts as long as fixing them did not involve altering requests belonging to other missions. In this mode it compiles a report of all of the actions taken and displays it to the user, with the option to revert any of the conflict resolution actions. AaCRONEM will leverage the mixed-initiative conflict resolution assistant functionality available from the AFSCN Scheduler code base. Figure 5 shows the results of this assistant in the Air Force satellite scheduling solution that AaCRONEM will be leveraging.

Figure 5: Result of the Mixed-Initiative Conflict Resolution Assistant

Multi-Objective Resource Optimizer

The multi-object resource optimizer will serve as a localized resource optimization strategy that focuses on taking a viable schedule (or viable sections of a conflicted schedule) and modifying it to improve mission satisfaction with the results.

The optimizer will have two primary components: a local optimization component and an analysis component. The local optimization component will have a great deal in common with the conflict resolution analysis, in that it will be trying different types of schedule modifications (many of which involve swapping elements, although constraint softening will be less prevalent). It is expected that they will share some common code modules. The difference is that, whereas the conflict resolution's options will be graded and then evaluated by the user, the optimization options will be evaluated internally by the analysis component.

The analysis component will be able to determine the quality of a request's assignment as well as the tradeoffs among different assignments (a related but more holistic analysis). It will be invoked to determine the order in which candidate optimizations should be tried, and to evaluate the success level of the optimization results. It may also be

invoked, as part of scheduling, to perform minor optimization during the primary scheduling process (at which point it is often easier to shift requests; a complete schedule is a complex and inter-related network with limited flexibility for optimization).

In the process of this search and analysis, the logic will build up a history of the optimization process that may be displayed to the user. This will allow the user to evaluate the impacts of the optimization, and potentially revert portions of it.

Execution, Monitoring and Repair Module

AaCRONEM may monitor the actual overall space network under the auspices of SCaN to determine if any modifications are necessary to the plan due to the network state being different than assumed during the planning phase. Using OPNET tools, AaCRONEM will be able to anticipate how the actual space network will react to current and near-term conditions. Furthermore, based on the planned knowledge of upcoming network demands and the current understanding of the network capacity, AaCRONEM will be able to determine if it would be prudent to modify/repair the schedule so that it will actually be able to execute successfully.

What-if module -- since OPNET Modeler is able to simulate the entire network and allow the user to modify the network itself, place different loads on the network, etc., many types of what-ifs become possible. This allows greater flexibility in validating plans before they occur.

This module will also deal with emergency scheduling or any other real-time change to the schedule (such as a launch slip), in these situations there is the need to perturb the existing schedule as little as possible. That is, only the scheduled communication events that are immediately impacted by the new high-priority communications due to the emergency or launch slip should have their schedules changed. This is because real-time schedule changes will take some effort, specifically for each change to notify the appropriate parties (the user and the managers of the affected ground stations and other resources).

Case-Based Reasoning

Case-Based Reasoning (CBR) is the field of AI that deals with the method of solving a current problem by retrieving the solution to a previous *similar* problem and *altering* that solution to meet the current needs. Case-Based Reasoning is a knowledge representation and control methodology based upon previous experiences and patterns of previous experiences. These previous experiences, or "cases" of domain-specific knowledge and action, are used in comparison with new situations or problems. These past methods of solution provide expertise for use in new situations or problems. Based on our extensive experience with planning and scheduling systems, we believe that this project is a natural application for CBR. In particular, CBR could be used to guide the system's actions by allowing it to suggest conflict resolution strategies either to customers or that it could apply itself.

3. INTEGRATION OF CURRENT NEN, DSN & SN SCHEDULERS INTO COMMON SCAN SOLUTION

Per NASA input and feedback, there are challenges beyond the significant challenge of scheduling the SCaN network. A successfully solution that will be willingly adopted across the NEN, DSN & SN communities will require the following:

- An open communications loop with all the communities. This will include learning the current scheduling solution, procedures and methodologies for each community.
- Determining, in consultation with each community and the SCaN office, what current capabilities should be accessed through interfacing with current systems. E.g., leveraging a current community's scheduling engine, then interfacing with these.
- In addition, a community's scheduling engine could be used in parallel with AaCRONEM's internal Aurora-based scheduling; this would provide for separate parallel scheduling algorithms to be simultaneously called, with their resulting schedules being dynamically evaluated.
- Learning how AaCRONEM will need to fit in the current data flow, i.e., determining what systems AaCRONEM will receive information from, and what systems it will output information to, then interfacing with these.
- Providing updates rapidly, so that in addition to major releases, minor releases and concepts will be shared with the communities to garner feedback important to directing our development.
- A prime example is the user interface; it is easy to make changes to it rapidly between major releases.
- Providing enhancements in a way that is seamless to current users. So if AaCRONEM provided everything currently available in a way that operates very similarly, and then offered enhancements desired by the users, then it will be accepted, if not demanded, by the users.
- Being as open as possible; making all interfaces both simple and fully documented. The goal is that NASA and others can manipulate the interfaces without needing to consult Stottler Henke.

The AaCRONEM architecture supports the above integration requirements or the way we have developed the Air Force AFSCN system shows that we follow these requirements, as they are also needed for the AFSCN system. For example, as mentioned above the AaCRONEM architecture includes the "Different Algorithm Scheduler", this could represent a NASA scheduler, such as the current scheduler used by the SN, this address the second bullet item. As shown above the interface is flexible so if SN users needed some differences in the UI versus DSN users this can be easily supported and maintained.

4. AaCRONEM Prototype

The AaCRONEM prototype demonstrates some central features of the software specification, using data combining assets in the Near Earth Network (NEN) and Tracking and Data Relay Satellite System (TDRSS). In the following sections we describe the method with which the prototype:

- Takes Space Communications and Navigation (SCaN) **model files as inputs**. These files specify satellite network resources and missions, communication requests, and constraints.
- Provides a **user interface for editing** the model specified in data files, and for performing scheduling.
- **Displays scheduling results** in assignment and conflict summaries, and navigable graphical schedule.
- Provides facilities for **mixed-initiative conflict resolution** in generated schedules.

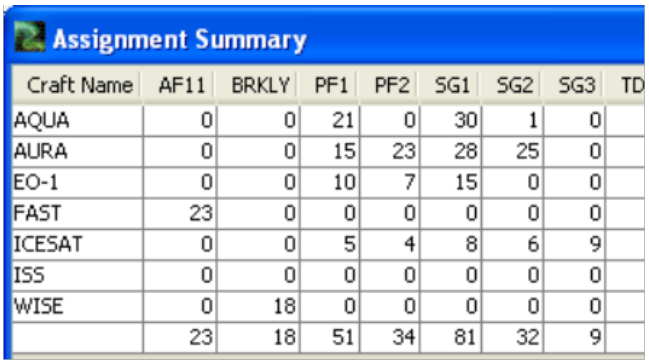
The example data used in the prototype includes a variety of NASA, DoD, and other U.S. governmental and international research satellites, in combination with TDRSS satellites. Network resources include the associated NEN ground stations, commercial providers, and TDRSS ground stations.

Scheduling and Results

After invoking scheduling using a UI button, the Aurora scheduling engine is employed to map mission communication requests to resources. Higher priority missions are scheduled first, allowing them to thereby satisfy their communication requests first. The Aurora post-processor performs pre-paid optimization, attempting to maximize pre-paid use and to move excess pre-paid communications to non-pre-paid satellites.

Results of scheduling are provided, including:

- An assignment summary of missions to ground facilities, see Figure 6.
- A conflict summary showing unresolved resource and temporal conflicts, if any. Conflicts may be shown per resource, per event, or as ordered within the schedule, see Figure 7.
- A navigable spatial allocation plot of resources relative to missions, see Figure 8.



Craft Name	AF11	BRKLY	PF1	PF2	SG1	SG2	SG3	TD
AQUA	0	0	21	0	30	1	0	
AURA	0	0	15	23	28	25	0	
EO-1	0	0	10	7	15	0	0	
FAST	23	0	0	0	0	0	0	
ICESAT	0	0	5	4	8	6	9	
ISS	0	0	0	0	0	0	0	
WISE	0	18	0	0	0	0	0	
	23	18	51	34	81	32	9	

Figure 6: Assignment summary

Figure 7: Conflict summary

The allocation plot shown in Figure 8 displays resources (e.g., PF1) relative to missions (e.g., AQUA), on a color-coded timeline.

- Lighter Blue – no access between PF1 and AQUA
- Darker Blue – other missions scheduled on PF1, or turnaround time for PF1
- White – open access relative to the AQUA
- Green – scheduled AQUA communication during an access period
- Red – conflicted AQUA communication

In the schedule there is a conflict indicated by the red portion of the timeline, which occurred because a request had to be scheduled when no access was available. Various means of resolution may be used to address the conflict.

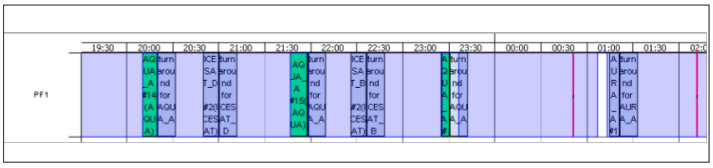


Figure 8: Allocation plot

Conflict Resolution

Conflicts normally occur because no schedule is possible without violating constraints. For instance, the stackup

conflict occurs when the end of the day is reached without all requests for that day being satisfied. Most conflicts must therefore be resolved by overriding constraints. Some conflicts can be resolved without relaxing constraints, for instance by moving a high-priority mission to a particular alternate valid spot, but conflicts usually arise because there are insufficient available resources to satisfy all requests.

Constraints may be overridden manually by changing schedule properties in the Aurora UI or graphically by clicking and dragging within the spatial plot, see Figure 9, which has the effect of changing constraints behind the scenes. Resource constraints may also be changed using the model editor. Any of those actions may invoke a rescheduling, which in turn may automatically resolve other conflicts. Figure 9 demonstrates an example where AQUA and AURA are scheduled for the same view period due to constraints, a situation resolved by clicking and dragging AURA to a new view period.

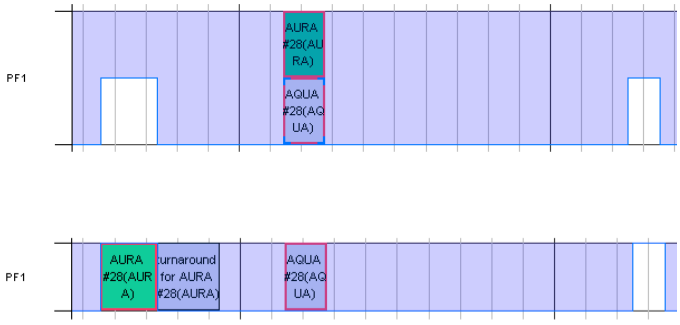


Figure 9: Before and after of a graphical constraint override in the spatial plot

Example scenario with NEN and TDRSS

The following spatial plots show an example of conflict resolution employing TDRSS. The schedule attempts to satisfy requests for the WISE satellite using only the BRKLY ground station. In this case, communication requests exceed the available access periods, and resource conflicts result, indicated by the narrow red boxes in Figure 10.

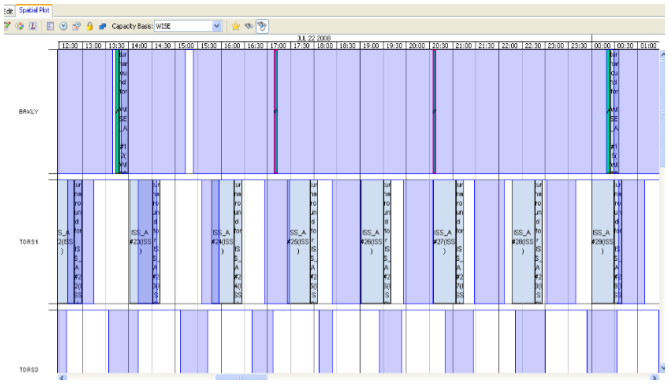


Figure 10: WISE satellite cannot complete the defined communications using its given ground stations

These conflicts can be resolved in this scenario by updating the antenna options to allow WISE to access TDRS1. This resolves most of the resource conflicts, see Figure 11

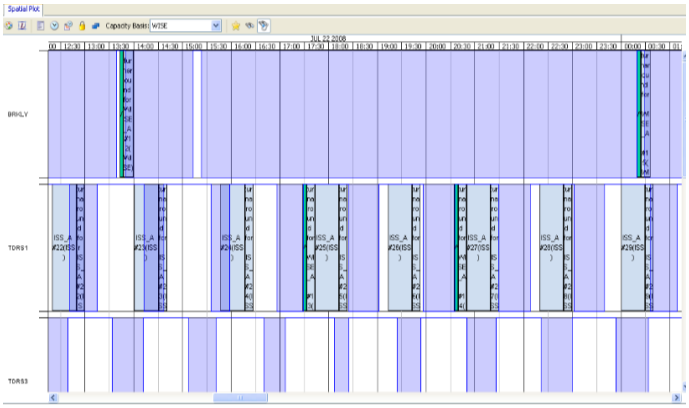


Figure 11: Conflicts resolved when WISE can access TDRS1 in addition to its own ground station

The AaCRONEM Prototype can solve quite complex situations already for both NEN & SN, either separately or in conjunction.

5. Conclusion

AaCRONEM will allow more-accurate modeling, support easier and more satisfactory conflict resolution, and result in more-satisfactory schedules for the missions. Then, during execution of the plan, AaCRONEM will monitor the situation, and adapt the plan and data routing to successfully complete a plan. AaCRONEM is based on Aurora and Aurora has outperformed every existing scheduling system in every domain in which it has been applied. AaCRONEM will be developed to be familiar to current users of the NEN, SN & DSN networks and thus feel like a natural extension of the tools they are already familiar with. AaCRONEM is also leveraging a project for the Air Force with similar goals for the Air Forces's satellite network. The process of creating this solution has begun, and includes the development of a prototype NEN/SN scheduler and deconflictor, which proved the automated scheduling/optimization/deconfliction capability of our approach

REFERENCES

[Barbulescu et al., 2006] L. Barbulescu, A. Howe and D. Whitley, "AFSCN Scheduling: How the Problem and Solution Have Evolved", Mathematical and Computer Modeling, 2006.

[Chien et al, 2000] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smit, F. Fisher, T. Barret, G. Stebbins and D. Tran, "ASPEN-Automating Space Mission Operations using Automated Planning and Scheduling", SpaceOps 2000, Toulouse, France, 2000.

[Clement and Johnston, 2005] B.J. Clement and M.D. Johnston, "The Deep Space Network Scheduling Problems", Proceeding of the Innovative Applications of Artificial Intelligence, 2005.

[Globus et al., 2003] A. Globus, J. Crawford, J. Lohn and A. Prior, "Scheduling Earth Observing Satellites with Evolutionary Algorithms", In International Conference on Space Mission Challenges for Information Technology, Pasadena, CA 2003.

[Gooley 1993] T.D. Gooley, "Automating the Satellite Range Scheduling Process", Master Thesis, Air Force Institute of Technology, 1993.

[Questus, web] Questus. http://questus.unitedspacealliance.com/

[Space Ops] "USA Develops Space Ops Toolset of the Future". Press Release. SpaceRef. (4 Apr. 2006). Online. 13 Aug. 2007 <http://www.spaceref.com/news/viewpr.html?pid=19433 >

BIOGRAPHY

Robert Richards, Ph.D. is the Principal Scientist and Manager of Stottler Henke's Navy helicopter training contract, OMIA. OMIA is a PC-based desktop training system that teaches crewmembers the Navy's new MH-60R and MH-60S helicopters. Dr. Richards has taken OMIA from a Research and Development SBIR project to a deployed training tool that has been awarded a $4.1 million IDIQ contract. Dr. Richards received his Ph.D. from Stanford University in mechanical engineering with an emphasis on machine learning and artificial intelligence. Dr. Richards is managing and has managed multiple projects for both commercial and government clients, including various intelligent-tutoring-system-based training projects. He is the principle investigator for VERTICAL, a Navy project to develop an innovative analytic test tool that can be used to support vertical takeoff and Visual Landing Aid analysis and testing. He was also the PI for INCOT, an Air Force project that developed automated tools for network layout. These projects exemplify his wide range of research and application area interests, including: training system development; applying automation and artificial intelligence techniques; and decision support tool development for life-critical situations. Dr. Richards has publications in all these areas.

Jeremy Ludwig, Ph.D. is a project manager / research scientist at Stottler Henke, where his research areas include intelligent training systems, machine learning, and behavior modeling. He is leading the software development effort for the OMIA common cockpit helicopter training system for the MH-60S and MH-60R. Dr. Ludwig has also been involved in a number of other research projects that utilize game and simulation technology for training. These projects include a mobile, game-based training system developed for the iPhone to support the training needs of F-16 and F-22 pilots, a game-based second language retention system also delivered on the iPhone, an on-line simulation-based training system with learning adversaries in a multi-player virtual world for desktop training, and SimVentive™, an integrated development environment for the rapid construction of training games and simulations. Jeremy was a co-chair for the 2008 AAAI Fall Symposium on Adaptive Agents in Cultural Contexts and has been involved in a number of tutorials on the use of artificial intelligence techniques in serious games at IITSEC over the past four years. He joined Stottler Henke in the fall of 2000 and holds a PhD in computer science from the University of Oregon.