The Use of Software Agents for Autonomous Control of a DC Space Power System

Ryan D. May
Vantage Partners, LLC.
Cleveland, Ohio, USA
ryan.d.may@nasa.gov

Dr. Kenneth A. Loparo
Case Western Reserve University
Cleveland, Ohio, USA
kal4@case.edu

***Abstract*—In order to enable manned deep-space missions, the spacecraft must be controlled autonomously using on-board algorithms. A control architecture is proposed to enable this autonomous operation for an spacecraft electric power system and then implemented using a highly distributed network of software agents. These agents collaborate and compete with each other in order to implement each of the control functions. A subset of this control architecture is tested against a steady-state power system simulation and found to be able to solve a constrained optimization problem with competing objectives using only local information.**

***Index Terms*—cyber-physical systems, energy management, microgrids, power system control, power system simulation, software agents**

I. Introduction

As mankind begins to develop spacecraft capable of moving out of the Earth-Moon system, the need for autonomous control of the spacecraft becomes critical. Past and current systems have relied on continuous support from an army of personnel at Mission Control and other locations for even typical day-to-day operation. Because communication latencies for all previous missions have been less than two seconds, this type of off-board control has been feasible. However, missions to Mars will have a communication latency that varies between approximately 6 minutes and 44 minutes round trip; this latency makes direct system control by ground support infeasible. In addition, the current approach places the astronauts in a very challenging position if there should ever be a loss, even momentarily, of communication.

The goal of developing a manned spacecraft capable of autonomous, or semi-autonomous, control is too broad in scope to tackle at once. Thus this work focuses on developing an approach to autonomously control the vehicle's electric power system (EPS) under the guidance of a vehicle Mission Manager. The mission manager coordinates all of the various sub-systems on the spacecraft, such as life support, thermal, communications, and power to best achieve the desired mission objectives given operational constraints. A simple version of this architecture has been demonstrated on the NASA Habitat Demonstration Unit [1], [2] and work towards developing a prototype for interfacing with an autonomous power system is currently ongoing [3]. Further complicating the picture is that many proposed future spacecraft are actually multiple spacecraft that can berth and unberth as needed to achieve mission objectives. When docked, the power systems of the vehicles can be connected into a single power system that must operate appropriately.

This work was funded by the NASA Advanced Exploration Systems Project's Modular Power Systems Task.

All of these design and mission concepts lead towards requiring a power system that is highly robust, adaptable, and reconfigurable in an autonomous or semi-autonomous manner. This paper will begin to develop a control system that can operate an electric power system to achieve these characteristics. To enable the desired plug-and-play capability as well as improve system robustness, the control system will be a decentralized system comprised of software agents that communicate in a peer-to-peer fashion.

This paper will present a functional hierarchy that can enable autonomous control of an electric power system. A subset of this architecture is then implemented using software agents that are then inserted into a MATLAB steady-state simulation of the power systems of two generic spacecraft. The two spacecraft share a power interconnect to allow the systems to exchange power when it is mutually beneficial. A simple mission manager determines the priorities of the loads on the two power systems and passes that information to the software agents. The agents then interact in a peer-to-peer topology to solve and implement an optimal power allocation problem.

The proposed functional control architecture of an autonomous space power system is discussed in Section II. A brief description of software agents is presented in Section III followed by a description of how agents can be used to implement the functions of the control system. Some initial simulation results will be given in Section V. Finally, Section VI will highlight conclusions and discuss future work.

II. Proposed Control Architecture

The proposed control architecture, shown in Figure 1, is inspired by the approach taken by terrestrial Regional Transmission Organizations. The architecture is split into four hierarchical layers based on the required time-to-respond of the particular operation. From bottom (fastest) to top (slowest) these layers are: the hardware reactive layer, the software reactive layer, the optimization layer, and the coordination layer.

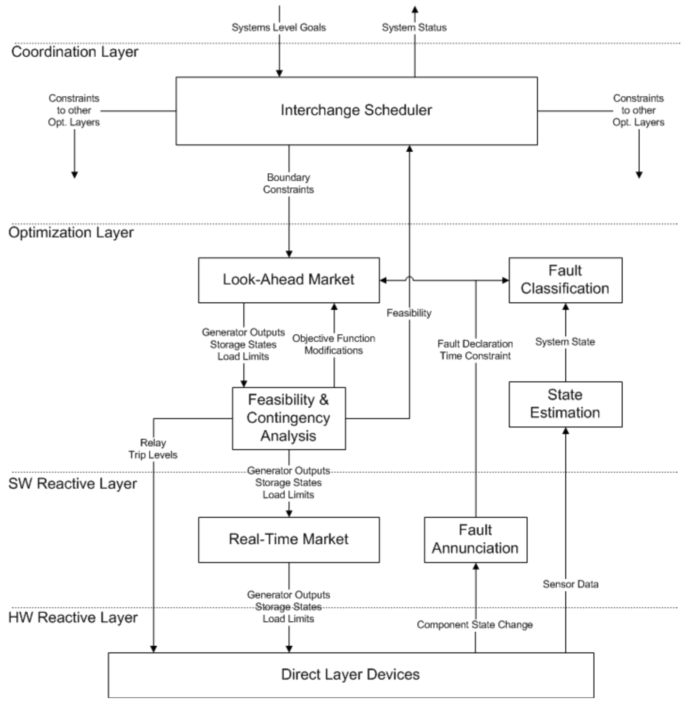

Figure 1. Proposed functional architecture for an Autonomous EPS Controller

The hardware reactive layer (also referred to as the direct control layer) is comprised of the fastest reacting devices in the system, typically the hardware protection system and all device-level control loops. Components include breakers, battery charging hardware, and voltage regulating devices.

The software reactive layer includes software or communication actions that do not need to coordinate with other devices in the control system. An example is a component declaring that a fault has been detected. This action would occur as soon as the detection logic determines that a fault has occurred; a much slower process than a breaker tripping, but likely much faster than a process that requires communication between various nodes throughout the system. Another example is that a hardware setpoint may be changed in response to new data (either via sensor or obtained via communication channel). Again, the decision to make the change in setpoint can be made locally without coordinating with others.

The optimization layer is the first layer in which collaboration between devices in the system is required, and thus, it can only respond to and solve problems on a longer time-scale (i.e. minutes). The functions in this layer need information beyond what is locally available in order to take the proper action and so collaboration is required.

The top layer shown here is the coordination layer which enables systems to work together when needed. Again, collaboration and communication is required and thus long time-scales are of interest. This layer will act to modify the goals, objective functions, or constraints of the functions in the optimization layer to enable cooperation between systems.

It is foreseeable that additional "coordination layers" may be added to the top of this architecture. For example, to control a "grid of microgrids" each of the microgrids would have at least one coordination layer to enable the microgrids to work together. If the "grid of microgrids" wanted to operate alongside another such system, then another coordination layer would be used to allow the two systems to operate together while each maintains their sovereignty.

The following subsections will discuss each of the functional blocks shown in Figure 1. For this work, the Interchange Scheduler, Look-Ahead Market, and a simple version of the Feasibility Analysis function are implemented and so are discussed in more detail.

Look-Ahead Market: The purpose of the Look-Ahead Market is to solve the power allocation problem for a future window of time. Generators and loads are required to predict their generation capacity/demand during the defined period of time. This information is then used to determine the output of each generator, the state of any network switches, state of all storage devices, power allocated to each load, and load trip points. In cases when there is not enough generation to meet the expected load, the loads with the highest priority will be served first.

It is easy to determine if a proposed solution is optimal. When there is not enough supply to meet the desired loads, the optimal solution will have the battery state-of-charge constraint active, meaning that the system will allocate the least amount of power possible to the battery to meet the SOC constraint. The optimal power allocation will give 100% of the desired allocation to the loads in priority order until there is no more power available. The same logic is true for the optimal supply allocation. When there is enough supply, the optimal solution is to provide power to all loads using the highest priority generators.

Feasibility and Contingency Analysis: The solution arrived at by the Look-Ahead Market needs to be examined to determine if it is feasible and secure. Security is determined by running a contingency analysis in which the system is simulated and component failures are introduced in order to determine if the system remains stable and within constraints immediately after the failure. If the system is not secure, then the Feasibility and Contingency Analysis block will modify the objective function used in the Look-Ahead Market to drive the next solution to be more secure. This loop will repeat until a secure solution is achieved or a time-out occurs, at which point the "best" solution found will be issued to the lower layer.

Interchange Scheduler: If the Feasibility Analysis determines that there is not enough generation capability in the system to supply the forecast load, the Interchange Scheduler will be notified. The Interchange Scheduler will then go to other Interchange Schedulers to determine if they have excess power that they can sell. If so, an agreement will be reached between the two Schedulers and the Look-Ahead Market in each system will be sent new boundary conditions (the scheduled power flow across the system boundary) from which to develop a solution.

Real-Time Market: The Real-Time Market is envisioned to be a function by which agents at the loads respond to signals from generators, storage devices, or network devices about small changes in power availability in order to account for small deviations from the predictions used in the Look-Ahead

Market.

State Estimation: The purpose of State Estimation is to use the available sensor data to determine the most likely system state that produced those measurements. This process can remove faulty sensor data, reduce measurement noise, and detect failed sensors.

Fault Annunciation: In the envisioned cyber-physical system, each device will be capable of detecting failures and faults of its own operation as well as unexpected input and output measurements. The device would then announce this fault so that the system can take the appropriate action.

Fault Classification: The Fault Classification function serves to take all of the fault announcements from devices across the system as well as the state estimation results and put them together to determine what failure or series of failures occurred. This information can be used to make more informed decisions as to how to restore the system to maximal capability.

III. Software Agents

The concept of software agents and multi-agent systems (MAS) were first formally described by Wooldridge in 1992 [4] and further expanded in 1995 [5]. Software agents are "an encapsulated computer system that is situated in some environment and can act flexibly and autonomously in that environment to meet its design objectives" [6]. They have well-defined boundaries and interfaces and are placed in an environment where they have, at least, partial controllability and observability. Crucially, agents have the ability to be proactive [5]. This capability differentiates them from the typical "object-oriented" model used in computer science. Objects are passive, and while they may include "behaviors," they do not have the ability to trigger those behaviors [7]. The other significant feature of a software agent is that they have the ability to interact with other agents through high-level social interactions [7]. In summary, a software agent exhibits features such as autonomy, social ability, reactivity, and pro-activeness. Agents may also exhibit other traits such as benevolence, rationality, mobility, or intelligence. Regardless of the specific traits, the key characteristic is the ability of agents to have "relationships" with other agents in the system. These agent-to-agent relationships are subject to change based on the operating situation, local environment, and design objectives. The agents use these interactions and relationships to achieve their design objectives through either cooperation or competition with other agents.

The fundamental challenge when designing a MAS is to develop a system capable of finding globally optimal (or near-optimal) solutions using agents with only local information. This requires coordination between the individual agents that is typically achieved through either competition or cooperation. Competition has typically taken the form of agents bidding in auctions [8], [9]. Various auction types and bidding rules from both economics and game theory have been investigated as a means to enable agent-based dynamic pricing systems [9]. Most of the agent-based control systems proposed and implemented utilize some manner of auction, but this is not necessary. Systems of cooperating software agents are found most commonly in robotic swarms. The most common formulation of inter-agent cooperation is to get a number of disparate agents to agree on a value of interest; this type of problem is known as a consensus problem [10], [11]. A classic manifestation of this problem is to enable a swarm of robotic agents to agree on a direction in which the flock should move. Other examples include vehicle formations, attitude alignment, rendezvous problem, coupled oscillators, and robot position synchronization.

Unfortunately, there is no generally accepted process that will lead to an optimal (or even a good) design. However, when developing MAS for solving control problems there are a number of good "rules of thumb" to follow:

- Software agents are not suited to tasks that require very fast response; they require time to collaborate/compete with other agents in the system to arrive at a solution [12]. Therefore it is often helpful to decompose the problem according to time-to-respond and delegate agent functionality at the appropriate levels.
- Functionally decompose systems into components that have similar objectives [7]. For example, a load in a power system may have the objective to secure a desired amount of energy over some specified time window. We need not be concerned if the load is a washing machine or a blast furnace, those differences result in different parameters (e.g. how much energy is desired, ramp characteristics) but they both have the same objective.
- Agents should be both active and autonomous; in particular, "individual components should localize and encapsulate their own control" [7]. This again emphasizes the desire to localize control as much as possible. As an example, an appropriate use for an agent would be to oversee a hardware implementation of a PID controller. The agent can modify the controller gains, but all high-bandwidth control decisions should be based on locally available data.
- MAS are ideal for large, complex systems where centralized control would be too "large" of a problem to solve. However, this complexity means that "it is impossible to *a priori* known [*sic*] about all potential links [between agents]: interactions will occur at unpredictable times, for unpredictable reasons, between unpredictable components" [7]. The designer must bear this in mind and not expect that communication will be timely, or the resulting system will be brittle when implemented. This quote is hinting at emergent behavior in the network, however if the agents are limited in scope then it is possible to ensure that communication only occurs between specific components. Further, proper scoping can ensure that the agent interactions occur with reasonably predictable randomness.

From an engineering design perspective it is critical to limit the scope and malleability of software agents. A completely flexible, constantly changing system would be impossible to test, debug, and validate. The agents should be designed to have only one goal/objective at any given time, or competing active goals may make the agent behave unpredictably.

IV. Agent-Based Implementation

For this initial work, the agent-based control system is built in MATLAB. The control system is comprised of five classes of software agents: load agents, storage agents, supply agents, network agents, and interchannel coordination agents. Due to page constraints, it is not possible to cover the details of how each of these agents is implemented. In general, each of the agents contains a significant amount of information about the device with which they are associated. For instance, a supply agent will have access to all data necessary to estimate the power output of the generator over any given time window. The agent also has the capability to control the device; for a load agent this could be limiting the energy consumption of the load to remain within the allocated level. Each agent also has access to all local measurements (current, voltage, etc) and the ability to control local actuators; however, no agent has system-level or global knowledge.

As stated earlier, only a subset of the total functional architecture was implemented for this study. Particular attention was paid to the Look-Ahead Market and the Interchange Scheduler as well as a simple Feasibility Analysis function. For this initial implementation, the Look-Ahead Market is an economic-based allocation mechanism. Each load agent submits a bid and each supply agent submits an offer for power during a ten minute window of time starting ten minutes in the future. These bids and offers are continuous functions that are monotonically non-increasing and non-decreasing, respectively, that can be easily summed and the intersection found to determine the economic optimal point. One advantage of this method over auction-type techniques is that there is no iteration required to arrive at a decision. Note that in order to ensure that control is truly distributed, each of the loads and supply agents submit the bid/offer to all connected network agents who sum the curves and pass the information along until all nodes have been covered. If a particular line is constrained to supply a maximum amount of power, the network agent on that line will modify the load bids to ensure that the constraint is observed. At the last network agent, the intersection of the load bids and supply offers will be computed resulting in the Market Clearing Price and the quantity of power cleared. This information will be returned to the loads, supply, and storage agents via the network agents. Each of these agents will use the information to determine how much power they have been allocated for the planned period of time.

After the market clears, any load that was not allocated the desired amount of power can issue a request to the interchannel coordination agent. This agent will take all the requests and seek to find another interchannel coordination agent with supply available at an acceptable price (i.e. the load bids and supply offer intersect). If one is found, a deal is made and the power cross-tie breaker is closed. The Look-Ahead Market is then rerun with the cross-tie power flow as a constraint on each of the systems. This constraint is implemented as new, temporary, load and supply agents that reside at the interchange connection point. These new agents will bid/offer the agreed upon power at the highest priority, ensuring that the agreement is fulfilled by both parties.

V. Simulation Results

A steady-state simulation of two generic spacecraft electric power systems has been developed in MATLAB (The Mathworks). As shown in Fig. 2, each system is independent, but has a cross-tie to enable the systems to share power when necessary. Each of the systems has a solar array to provide power during periods of sunlight (insolation) and a battery to provide power during eclipse. Each of the loads is a constant

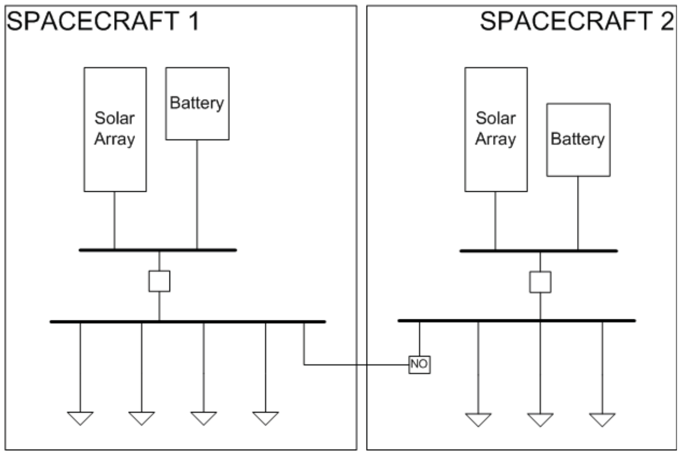

Figure 2. Schematic of the two independent, cross-tied, electrical power systems modeled in MATLAB.

power load with a priority as assigned by a simplified Mission Manager. The solar arrays experience periodic eclipses; each orbit is 90 minutes long with a 35 minute long eclipse.

In order to evaluate the control algorithms, Vehicle 1 is heavily loaded and the other is lightly loaded. The system is simulated for 600 minutes, during which the control system responds to the orbit. The solar array output and battery SOC for each vehicle are shown in Fig. 3. The control system successfully meets the SOC constraint by shedding low priority loads as shown (for a selected time window during insolation) in Fig. 4. The vertical axis in the Fig. 4 plots is the load allocation as a percentage of the load desired for each load and battery. As stated earlier, the battery's storage agent determines how much power it would like within charging limit constraints and bids for that amount at a priority level which is dependent on the remaining time until eclipse and the current state of charge. The top plot highlights the fact that Vehicle 1 is overloaded, and in order to meet the SOC requirement, the storage agent must increase its priority, driving other loads offline. Loads 3 and 4 are not allocated any power, and there are periods of time when Load 2 is only allocated 75% of the desired level so that the battery can charge at the maximum rate.

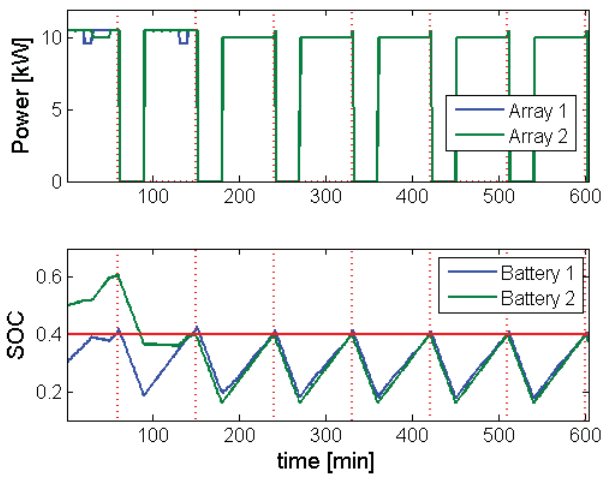

Figure 3. Solar array output and battery state-of-charge for both vehicles. The vertical red lines mark the start of eclipse.

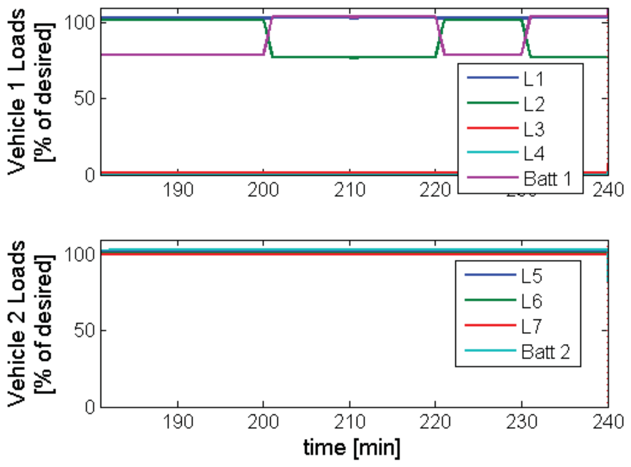

Figure 4. Example load allocation when the interconnection breaker is open.

Because of how the Interchange Scheduler function is constructed, each channel will operate independently and seek additional power as necessary. For the time window above, all of the power in Vehicle 2 has been allocated internally (6kW to the loads and 4kW to the battery) and thus there is no more to “sell” to other channels. During the eclipse periods, Vehicle 2 sends power to Vehicle 1. Since Vehicle 2 has power available after supplying all of its internal loads (a total of 6kW), it is clear that this is a good strategy.

For this situation, the solution of the power allocation problem (as described in Section II) is optimal. The two batteries obtain the minimal amount of power required to ensure that they do not violate the SOC constraint. The highest priority loads on each channel are served first when there is a shortfall in power availability. Finally, the channels cooperate to exchange power when one channel has a deficit and the other a surplus.

VI. Conclusions

There has been, and will continue to be, considerable interest in the application of software agents to distributed control problems, particularly in large complex systems such as power distribution systems. Unfortunately, many of the previously proposed solutions are not truly distributed and retain much of the centralized control found in modern systems. In this work, a control system functional architecture is proposed that can be implemented using truly distributed software agents. A subset of this architecture was implemented in MATLAB and integrated with a quasi-steady state simulation of two independent but coupled, generic spacecraft power systems. The control was tested and found to be capable of balancing multiple goals (provide maximal power to loads and ensure that the batteries are sufficiently charged) while being constructed in such a way that the amount of data communicated between the agents in minimized.

It is clear that much work remains to be done in this area. The remainder of the architecture presented here needs to be implemented and tested. For future work it is critical that a more capable and detailed electrical power system simulation be utilized in order to more accurately determine the control effectiveness.

Acknowledgment

The authors are grateful for the support of Jim Soeder, Ray Beach, Nancy McNelis, and Pat George of the NASA Glenn Research Center.

References

[1] D. Lawler, L. Wang, T. Ngo, L. Moreland, D. Carrejo, A. Schram, T. Matthews, C. Russell, E. Alex, and A. Qaddumi, “Habitat demonstration unit core avionics software,” *Johnson Space Center Beinnial Research Report*, pp. 138–139, 2011.

[2] S. A. Howe, K. J. Kennedy, T. R. Gill, R. W. Smith, and P. George, “NASA habitat demonstration unit (HDU) deep space habitat analog,” in *SPACE Conferences & Exposition*. American Institute of Aeronautics and Astronautics, 2013. [Online]. Available: http://dx.doi.org/10.2514/6.2013-5436

[3] R. D. May, J. F. Soeder, R. F. Beach, P. J. George, J. D. Frank, M. A. Schwabacher, , S. P. Colombano, L. Wang, and D. Lawler, “An architecture to enable autonomous control of spacecraft,” *to be published in the Proceedings of the IEEE EnergyTech 2014*, 2014.

[4] M. J. Wooldridge, “The logical modelling of computational multi-agent systems,” Ph.D. dissertation, University of Manchester, August 1992.

[5] M. Wooldridge and N. Jennings, “Intelligent agents: Theory and practice,” *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[6] M. Wooldridge, “Agent-based software engineering,” *Software Engineering. IEEE Proceedings*, vol. 144, no. 1, pp. 26 –37, Feb 1997.

[7] N. Jennings and S. Bussmann, “Agent-based control systems: Why are they suited to engineering complex systems?” *Control Systems, IEEE*, vol. 23, no. 3, pp. 61 – 73, June 2003.

[8] S. Bussmann and K. Schild, “Self-organizing manufacturing control: An industrial application of agent technology,” *Proceedings of the 4th International Conference on Multi-Agent Systems*, pp. 87–94, 2000.

[9] P. Wurman, “Dynamic pricing in the virtual marketplace,” *Internet Computing, IEEE*, vol. 5, no. 2, pp. 36–42, Mar/Apr 2001.

[10] W. Ren, R. Beard, and E. Atkins, “A survey of consensus problems in multi-agent coordination,” in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1859–1864 vol. 3.

[11] R. Olfati-Saber, J. Fax, and R. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[12] S. Amin and B. Wollenberg, “Toward a smart grid: power delivery for the 21st century,” *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, 2005.