

# SPACE TELECOMMUNICATIONS RADIO SYSTEM (STRS)

## COGNITIVE RADIO

Louis M. Handler (NASA Glenn Research Center, Cleveland, Ohio, U.S.A.;  
Louis.M.Handler@nasa.gov); Janette C. Briones (NASA Glenn Research Center,  
Cleveland, Ohio, U.S.A.; Janette.C.Briones@nasa.gov)

### ABSTRACT

Radios today are evolving from awareness toward cognition. A software defined radio (SDR) provides the most capability for integrating autonomic decision making ability and allows the incremental evolution toward a cognitive radio. This cognitive radio technology will impact National Aeronautics and Space Administration (NASA) space communications in areas such as spectrum utilization, interoperability, network operations, and radio resource management over a wide range of operating conditions.

NASA's cognitive radio will build upon the infrastructure being developed by Space Telecommunications Radio System (STRS) SDR technology. This paper explores the feasibility of inserting cognitive capabilities in the NASA STRS architecture and the interfaces between the cognitive engine and the STRS radio. The STRS architecture defines methods that can inform the cognitive engine about the radio environment so that the cognitive engine can learn autonomously from experience and take appropriate actions to adapt the radio operating characteristics and optimize performance.

### 1. INTRODUCTION

Cognitive radio is defined such that the radio senses the environment, understands context, acts on the knowledge, and learns from previous actions and results. If NASA uses cognitive radios, they should be able to communicate with both legacy and other cognitive radios because of the difficulty of replacing currently deployed radios in space. Common cognitive radio architectures are desired over a large class of radios so that the knowledge base and lessons learned may be used on the next radio thereby saving money by not having to reinvent such a radio each time. There may also be economies of scale so that the systems of radios built to the architecture pass information back and forth to optimize the system of radios rather than just a single radio.

### 2. STRS COGNITIVE RADIO EVOLUTION

If NASA uses cognitive radios in space, the requirements would differ from those for the usual terrestrial requirements in that satellites move rapidly, encountering a wider range of atmospheric, solar, and cosmic effects, normally using higher

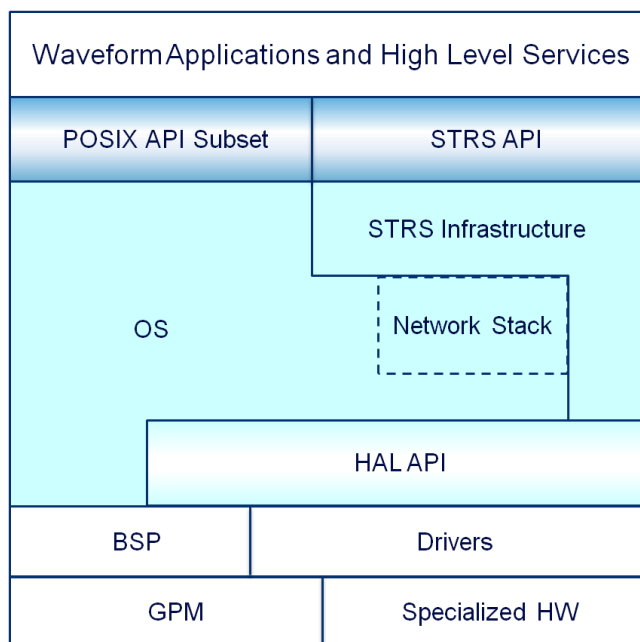


Figure 1 STRS Layer Cake Model

frequency waveforms, and with longer distances and delays between radios. STRS was designed with NASA's needs in mind and is an ideal base to extend into the cognitive realm.

To optimize data throughput within a system of cognitive radios including space-based radios, dynamic spectrum and resource management may be needed. For NASA, cognitive radio is not just concerned with frequencies and bandwidth, but may also be concerned with such things as power management, antenna direction, temperature compensation, or even switching waveforms.

The STRS architecture contains methods to query to obtain information about the SDR, its environment, and its waveform applications as well as methods to control the operation of the SDR. A cognitive engine may use these features to optimize performance autonomously under adverse conditions. The cognitive data for a NASA SDR must include information about mission requirements and radio capability; for example, a NASA SDR could control satellite navigation or antenna pointing.

A layer cake view of the STRS architecture is shown in figure 1. The waveform applications and high level services are shown as a software layer at the top with optional portions in specialized hardware such as field-programmable gate arrays (FPGAs) in the bottom layer. The initial idea is

to treat the cognitive portion of the radio as a high level service before breaking it down further.

In STRS, the waveform applications and high-level services are separated from the operating environment (similar to core framework) by application programmer interfaces (APIs) rather than some middleware. The operating environment manages the components of the SDR and consists of the STRS infrastructure that implements the STRS APIs as well as the operating system (OS) that implements the POSIX API subset, a hardware abstraction layer (HAL), and any board support package (BSP) and drivers needed to use the general-purpose processing module (GPM) or specialized hardware.

The cognitive process of Joe Mitola [1] is broken down into components that observe, learn, orient, plan, decide, and act. These parts of a cognitive engine may be separate or integrated with the software defined radio, but to maintain the basic STRS architecture, they will be considered to be separate. Another breakdown of the cognitive process is described in [2] where the software defined radio is put in its place as an independent entity controlled by intelligent agents or decision-making process with a radio-domain adaption layer between. In [3], kinds of information are categorized as link state, application state, spatial state, and environment state from which data is extracted for learning. There may be additional states as the mission may require.

There are a variety of ways to depict the radio of the future. Some of these abstractions use different notation and terminology to accomplish the same thing. Similar to the egg model in [2], figure 2 depicts a cognitive radio showing concentric layers and combining many features of these complimentary approaches. Figure 2 shows a cognitive radio that can perform its functions as a radio, obtain parameters defining internal state and external environment, make decisions, and implement those decisions.

This model shows the outside world as the outer layer with the radio as both a sensing and communicating device, controlled from the decision-making and learning core. The radio API may be any architecture such as STRS or Software Communications Architecture (SCA) with the Radio Adapter layer calling the appropriate radio-domain functions for that architecture. This model shows some of the pertinent STRS methods described in [4]. This model does not break down the radio functions, the decision-making, or learning cycles enough to get to the beginnings of an implementation.

In what follows, the integration of the SDR with the cognitive engine as shown in figure 2 is developed further.

### 3. LEARNING INFORMATION AND DECISION-MAKING

The outcome of the decision-making process may change as the context evolves. As described in [3] different categories of state information are kept for use by its Context-Aware Routing Engine (CARE) decision-making program. The information in the categories listed below is tracked information that is being observed through the radio and learned by the cognitive engine where actions are decided and acted upon by the radio to affect that information. The

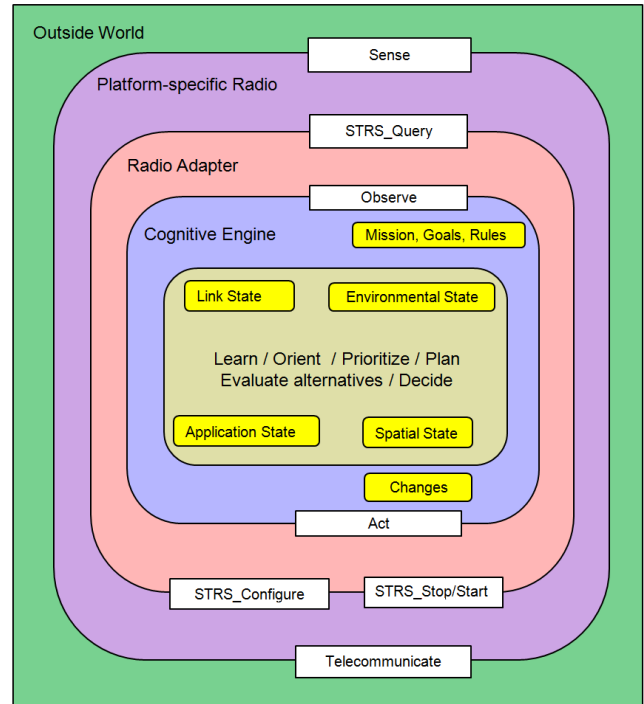


Figure 2. Combination Model of Cognitive Radio.

following sections have been augmented for additional state information that might be required for a space implementation. STRS provides the methods to obtain the state information required and to perform the actions required.

#### 3.1. Link State Information

Link state information includes such data as current channel bandwidth, observed bit error rate (BER), round trip time, transmit power, and the encoding overhead of the current modulation scheme. This link state information comprises a set of metric parameters that are platform independent and can be used for the evaluation of radio performance. Link state information can be captured from different sources. The current bandwidth and observed bit error rate (BER) can be most easily obtained from the running waveform application. Other link state information may be observed by a set of services that monitor various sensors and external signals.

#### 3.2. Application State Information

This is information about the currently running waveform applications and the types of data they wish to send. This information includes carrier frequency, modulation scheme, symbol rate, etc. Different types of data are sensitive to changes in particular network characteristics, and these sensitivities must be considered when making routing decisions. For example, real-time traffic may be tolerant of loss, but less so of jitter; vehicle health telemetry may require low bandwidth, but have bounds on latency; and large data files from science instruments may need high bandwidth and low error/loss rate, but may accept some

latency or jitter. In situations where multiple types of data are awaiting transmission, it may be determined that it is feasible to transmit one type given the current environment, while determining that no viable path exists for another type. The application state can be obtained from the waveform application.

### 3.3. Spatial State Information

Information about the relative physical locations of other nodes in a network may be available. This information can be used to help determine the best of several possible neighbors to use as a relay. Additionally, if the nodes move along well-known paths (for example, a satellite in orbit), then the future location of neighbors can be determined. In this case, transmission of data may be deferred in anticipation of a known-good relay becoming available. The contact graph routing mechanism proposed as part of the Delay-Tolerant Network (DTN) framework formalizes this concept for end-to-end path computations when future node positions are known but there is no connected path. Spatial state can be obtained from signal round-trip times, time of day, orbital computation, and known relationships. Global Positioning System (GPS) provides a way to determine location that is an element of spatial state.

### 3.4. Environment State Information

This includes information about the current local environment of the node, such as available electrical power, temperature and other health information. If power consumed by the radio is a concern, the cognitive radio may decide to relay data through a neighbor that can be reached at lower power instead of one that may provide a better end-to-end path. Temperature information may be used to adjust application parameters for optimal performance for that temperature. Environment state information can be obtained by a set of services that monitor various sensors and external signals.

### 3.5. Radio Platform Information

In addition to the states delineated in [3], the radio platform information must be kept for use in the decision-making process. This information includes the configuration parameter set of the radio platform to carry out required waveform and link operations. This configuration is platform specific and the settings depend on the required radio functionalities.

### 3.6. Mission Information

Beyond the state is the mission definition in terms of what has to be done, with what kind of radios, where, and when. Since NASA missions usually are concerned with collecting data, the source of that data may only be available when the radio is in a certain range of positions. Also the capability and reconfigurability of the current software-defined radio must be specified. There may be fixed rules as well as learned rules.

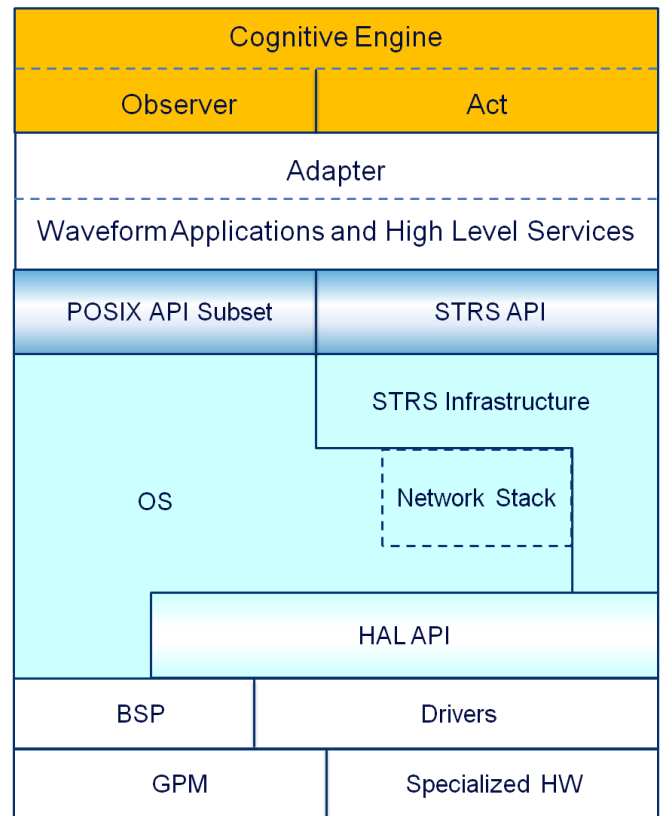


Figure 3. STRS Integration Layer Cake Model.

### 3.7. Cognitive Process

The current state, known limitations, historical trends, schedule, and mission requirements must be captured in one or more databases. In [1] Radio Knowledge Representation Language (RKRL) is suggested and in [3] Web Ontology Language (OWL) is suggested for a radio-oriented ontology using extensible markup language (XML). Since eliminating an XML parser was one of the defining features of STRS, we will study whether another standard data format can be recognized and used by a decision-making process.

The learning process should include saving data as a function of time because many data items may have start and stop times, or may be cyclical in either space or time. Decisions may be made according to some detected event, by the gradual fade in or out of some signal, or by a predicted scenario. A predicted scenario may be used to approximate when events will occur to optimize computational effort of the decision-making process.

Once the decisions are made, actions are determined. Possible actions may include changing path selection, setting quality of service parameters to meet application needs, deferring transmission until a later time (possibly by passing the traffic to a local DTN), transmitting some types of traffic ahead of others, requesting a particular encoding from the underlying cognitive radio, notifying applications of the current context so that the applications can make decisions about what data to send next, or some combination of these.

#### 4. HIGH LEVEL DESIGN

Figure 3 shows the high level layer-cake design of the STRS with cognitive capabilities. The waveform applications and services use STRS-specific APIs to call the STRS infrastructure as well as a POSIX application environment profile (AEP) to call functions within the OS. Similarly, the STRS infrastructure uses an STRS-specific API to call the waveform applications and services.

The cognitive engine may be thought of as an extension of one or more services such that its connection to the STRS software defined radio is through the standard APIs. The cognitive engine is further broken down internally as shown in [1]. A general cognitive engine obtained separately would have components for observe and act that have a different interface from the STRS APIs, in which case an additional radio adapter layer would be required.

In figure 3, an adapter layer is added to the high level services as shown in figure 1. Also, in figure 3, the cognitive engine is added at the top with its connection to the adapter layer confined to the functions of observe and act. The radio adapter interface consists of an observer part that is responsible for sending platform knowledge to the decision-making process, and an actor part that uses configuration and control information to instruct the radio to perform specific operations by changing parameters or applications to improve the functioning of the radio when dealing with situations not planned initially.

#### 5. STRS COGNITIVE RADIO DESIGN

A unified modeling standard (UML) design is shown in figure. 4. The UML provides a simple object oriented design in which we can define data structures as well as data content. The Observer passes the data to the cognitive engine in a format that it can use for processing. The Observer may obtain data at various time intervals, prepare, and save the data for the cognitive engine. The Observer would not need to test the data. The data would be time stamped and stored for learning and testing.

The Actor obtains data from the cognitive engine for controlling the radio.

The cognitive engine in figure 4 is a placeholder for all the artificial intelligence that makes it a cognitive radio. The cognitive engine should be in control of the radio, deciding whether there is a problem, determining the alternative actions, selecting which one is appropriate, and performing the most appropriate action. The Decision-Maker should have various kinds of tests and values, but it also needs rules, heuristics, priorities, and actions.

The Observer is broken down further by specifying an associated platform-specific radio-adapter class, which is denoted by ObserveSTRS. ObserveSTRS is the part of the radio-adapter that uses STRS APIs to obtain data, which it then reformats for the Observer portion of the cognitive engine.

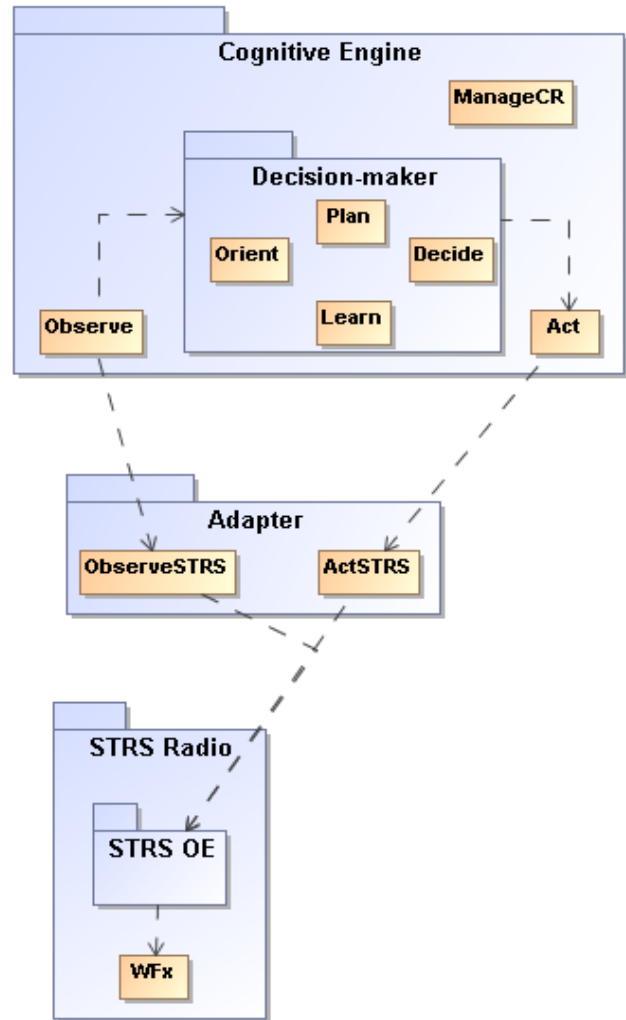


Figure 4. STRS Cognitive Radio

Similarly, the Actor is broken down further by specifying an associated platform-specific radio-adapter class, which is denoted by ActSTRS. ActSTRS is the part of the radio-adapter that translates the actions from a format that the cognitive engine knows to a format that the radio can use. The ActSTRS uses the SDR's APIs to change the function of the radio to carry out the decisions.

The Cognitive Engine in figure 4 could be broken down further into blocks to orient, plan, decide, learn, etc. The Cognitive Engine should evolve from one that performs tests on variables to determine when the radio is in a fault state with the simple action to reboot the radio to one that is truly cognitive; i.e. that changes parameters or applications to improve the functioning of the radio alone or in a system of radios.

In a well-understood environment, a knowledge-based system can make excellent decisions and provide the appropriate adaptation. But in a rapidly changing space environment, subject to varying temperature and radiation effects, or one subject to long delays due to large distances

between radios, the radio must be able to act more autonomously.

In the evolution of the cognitive engine, the orient and plan functions could be combined when the testing that establishes priority or severity cannot be separated from some knowledge of what is wrong or could be improved with some corresponding action. But, more likely, each component would be broken down, adding additional classes or packages, to specify its functionality in greater detail.

As part of the learning process, for example, one would likely add blocks for data mining, neural networks, etc. These are just a sample of what artificial intelligence modules might be necessary. Learning systems require observations to draw conclusions. Neural Networks (NN) can be trained by providing a set of known inputs and desired outputs, and it is these observations that are used by the NN to learn the desired behaviors. Feedback provided by monitoring the results of using the outputs, is required for learning.

The stored data or functions of the data could be mined periodically for interesting trends or recurrences that could be taken advantage of. A predictor would know the schedule of events along with any location information, possibly including satellite ephemeris data.

The added ManageCR component in figure 4 is a placeholder for the meta-learning process. This component controls the learning process and tries to improve that learning process. For example, on long missions during times when nothing much is happening, time and power may be available for further data mining whereas during normal operation, some of that time and power may be needed for transmission or reception.

When a detailed design is created, both observed data and inferred data would have to be added to figure 4. A certain amount of current and historical data must be kept for learning. In the simplest model they could be stored in tables or simple files. As greater complexity is added, one or more databases would probably be required.

The observed data may be supplied by querying the operating waveforms, the operating environment (OE), or any services that supply any additional information. For example, a service could be written to supply information about what external radio signals are encountered, their frequency, and signal strength that would be tested against the predicted schedule to determine when a new satellite comes over the horizon. To allow a fairly general architecture to be used, much of the rest of the data should be in configuration files.

The following must be considered in implementing the configuration files:

- 1) Allow decisions based on complex equations involving values of multiple attributes over multiple applications.
- 2) Allow decisions based on experience/learning.
- 3) Allow decisions based on data mining, neural networks, etc.
- 4) Allow decisions based on evaluating prioritized alternatives.

- 5) Allow a more complicated range of actions than merely changing a parameter.
- 6) Include a bridge that allows different cognitive architectures to play with different radio architectures.

## 6. CONFIGURATION FILE DATA

To keep the design and implementation as general as possible, the data items for the cognitive functions should be defined in configuration files. Since the cognitive functions don't have to be STRS applications, the configuration files may or may not follow the same format as the STRS application configuration files. There are specific formats required by various commercial off-the-shelf (COTS) products that perform corresponding cognitive functions. Therefore, there may need to be additional interface pieces to convert the formats.

### 6.1. Observer/ObserveSTRS data used

The observed data should be obtained using STRS\_Query or STRS\_RunTest for each active application and any services that monitor the platform or external sensors. To make this data configurable, the name in the Decision-Maker would correspond to a process for obtaining the data from the SDR. A configuration file would contain:

- 1) The name of the item in the Decision-Maker
- 2) The name of the attribute in the SDR (parameter name)
- 3) Where in the SDR to find it (the component's handle name in the SDR)
- 4) How to find it in the SDR (STRS\_Query or STRS\_RunTest)
- 5) Test ID if STRS\_RunTest is used
- 6) How often to check for changes or whether event driven

### 6.2. Decision-Maker data used

The Decision-Maker data should have various kinds of tests and values for that test. It should have rules, priorities and actions to be performed when a problem exists. A configuration file would contain:

- 1) The type of test or equation or heuristic to perform
- 2) The values needed for that test, which may include names of data items observed or learned
- 3) Actions to take if the test shows a problem
- 4) Priority/severity for each problem

### 6.3. Actor/ActSTRS data used

Actions requested by the cognitive engine must be translated into radio commands. For example, actions to reconfigure some parameter may be translated into an STRS\_Configure command or may require a preceding



STRS\_Stop and following STRS\_Start. A configuration file for actions would contain:

- 1) The name of the item in the Decision-Maker
- 2) The name of the attribute in the SDR (parameter name)
- 3) Where in the SDR to find it (the component's handle name in the SDR)
- 4) Action to take in the SDR as a combination of the following:
  - a) STRS\_Configure
  - b) STRS\_Stop
  - c) STRS\_Start
  - d) STRS\_AbortApp
  - e) STRS\_InstantiateApp
  - f) STRS\_Initialize
  - g) STRS\_Reboot (new) or equivalent
- 5) Configuration file to use if the application is reinitialized
- 6) Value of a corresponding attribute to change in SDR

Although the first three items in the list above appear to be the same as the first three items in the list for the observer, they do not have to be the same since there may be attributes that are queryable only (not configurable) as well as actions without attribute values to configure.

## 7. FUTURE EFFORTS ON COGNITIVE RADIO

When designing an STRS cognitive radio for a particular purpose using the STRS architecture standard [4], the user must understand the capabilities of the radio, the mission requirements, the allowed actions, and supporting data. Then, the user must design how the data will be kept, decisions made, and actions performed. The cognitive engine may be purchased and integrated into the radio or developed separately. The cognitive engine will use the

information learned and the available actions to select the appropriate actions.

It is assumed that not every cognitive radio function can be anticipated; however, a framework should be established that would encompass all types of available actions and use any internal or external data to decide what to do and cause the appropriate actions. Such a general framework is suggested in the previous section. The design indicated in this paper will allow technology improvements to optimize radio performance, using the latest state-of-the-art cognition engine. The performance feedback allows both the creator of the waveform application and the radio itself to determine changes needed in the variables observed and actions performed, optimizing radio performance and autonomous decision-making for SDRs under space and ground conditions.

The integration of artificial intelligence and SDR technologies enables the new field of cognitive radio. These enabling technologies will create new capabilities for new science opportunities. Cognitive radio technology has the potential to lower the operational costs and improve performance when included in NASA's existing networks.

## 8. REFERENCES

- [1] J. Mitola and G. Maguire, "Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, Vol 6, No. 4, August 1999.
- [2] B. Le, T. W. Rondeau, and C. W. Bostian, "General Radio Interface Between Cognitive Algorithms and Reconfigurable Radio Platforms," *SDR Forum*, 2007.
- [3] W. D. Horne, T. Suaris, R. T. Gilstrap, and R. Rogalin, "Developing the Building Blocks for Cognitive Communications: Adaptive Rates & Intelligent Networking," *IEEE Aerospace Conference*, 2011.
- [4] Space Telecommunications Radio Systems (STRS) Architecture Standard, Version 1.02.1, NASA/TM-2010-216809.