Achieving Fast Operational Intelligence in NASA's Deep Space Network Through Complex Event Processing

Joshua S. Choi[1], Rishi Verma[2], and Shan Malhotra[3]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

NASA's Deep Space Network (DSN) is a complex, global project, in which the expertise of human operators remain crucial for its successful operation. To find ways to save costs in operations and to improve its services, a number of modernization efforts are underway in the DSN. One such effort is a research and technology development task at the Jet Propulsion Laboratory that is investigating the use of *complex event processing (CEP)* for intelligent assessment of situations, trend analysis, and advanced automation. The technology leverages the significant business intelligence (BI) and data science advancements made in the enterprise industries over the last several years. The open source big data processing engine Apache Spark™ and the high-throughput, distributed messaging system Apache Kafka form the core of the DSN Complex Event Processing (DCEP) framework. This paper discusses the system engineering perspective of why achieving efficient, lower-cost operations in the DSN is a challenging problem, how the DCEP system handles the use cases that help realize intelligent operations, and how this solution fits into the overall model of the planned DSN Follow-the-Sun Operations (FtSO).

Nomenclature

3LPO	=	three links per operator
API	=	application programming interface
BI	=	business intelligence
CEP	=	complex event processing
DCEP	=	DSN Complex Event Processing
DR	=	Discrepancy Report
DSCC	=	Deep Space Communications Complex
DSN	=	Deep Space Network
DSS	=	Deep Space Station
DRMS	=	Discrepancy Reporting Management System
FtSO	=	Follow-the-Sun Operations
GUI	=	graphical user interface
JMS	=	Java Message Service
JPL	=	Jet Propulsion Laboratory
JSON	=	JavaScript Object Notation
LCO	=	Link Control Operator
LTPS	=	Light Time Physics Service
MCIS	=	Monitor and Control Infrastructure Services
MDS	=	Monitor Data Service
MDR	=	Master DR
MON-2	=	DSN Monitor and Control Standard
NASA	=	National Aeronautics and Space Administration

[1] Engineering Applications Software Engineer, Mission Control Systems Section, 4800 Oak Grove Drive, M/S: 301-480, Pasadena, CA 91109-8099, United States of America.
[2] Scientific Applications Software Engineer, Instrument Software and Science Data Systems Section, 4800 Oak Grove Drive, M/S: 158-242, Pasadena, CA 91109-8099, United States of America.
[3] Engineering Applications Software Engineer, Planning & Execution Systems Section, 4800 Oak Grove Drive, M/S: 301-250D, Pasadena, CA 91109-8099, United States of America.

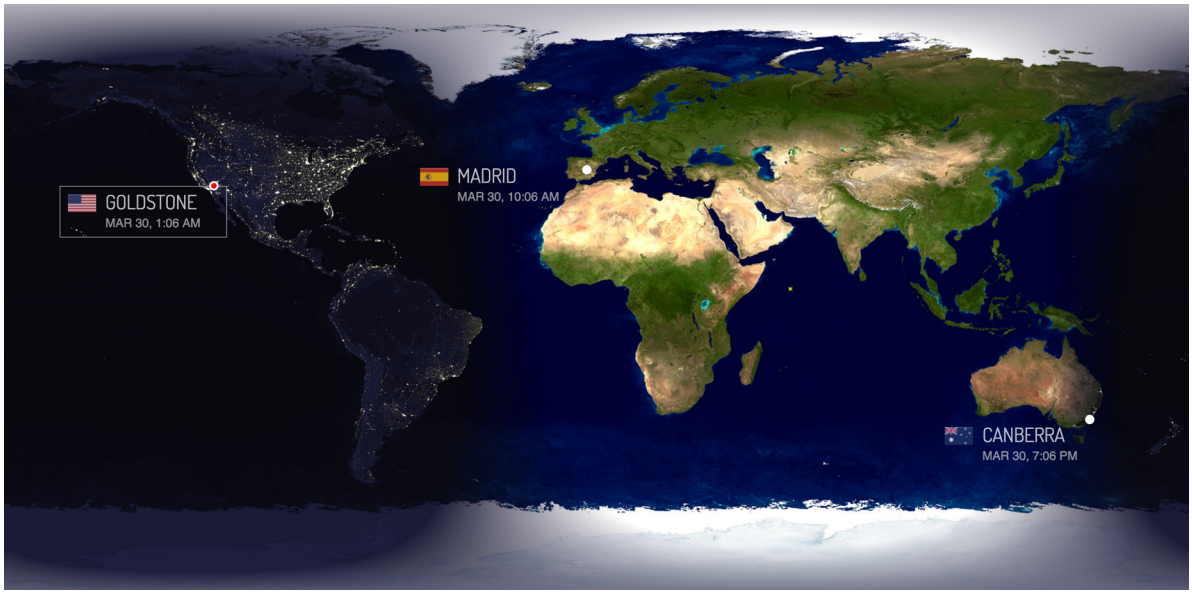

Figure 1. DSN's DSCC locations around the world. By dividing the circumference of Earth into approximate thirds, the DSCCs have a view to any spacecraft in deep space at any point in time, even as the Earth rotates. The three DSCC locations are: Goldstone, California, USA; Madrid, Spain; and Canberra, Australia. (Image credit: Deep Space Network *Now*; http://eyes.nasa.gov/dsn/dsn.html)

NMC	=	Network Monitor and Control (subsystem)
NOCC	=	Network Operations and Control Center
P_c/N_0	=	carrier power to noise spectral density ratio (signal-to-noise ratio)
PL/SQL	=	Procedural Language/Structured Query Language
REST	=	representational state transfer
RO	=	remote operations
SOE	=	Sequence of Events
SPS	=	Service Preparation Subsystem
SQA	=	Service Quality Assessment (subsystem)
TDN	=	Temporal Dependency Network
WAN	=	wide area network
XML	=	Extensible Markup Language

I. Introduction

NASA'S Deep Space Network (DSN) is a global network of space communication facilities and powerful antennas capable of communicating with interplanetary spacecrafts. Operated by the Jet Propulsion Laboratory (JPL) for NASA, at any given time, between 40 to 50 space missions rely upon its services. The DSN consists of three Deep Space Communications Complexes (DSCCs) around the world, geographically spaced apart by approximately 120 degrees on Earth for uninterrupted view of any spacecraft in deep space (see Figure 1). The DSCCs are operated onsite and—for the most part—independent of each other. Global mission support activities in the DSN are coordinated and monitored from the Network Operations and Control Center (NOCC) at JPL in Pasadena, California. At every DSCC and also at the NOCC, numerous heterogeneous hardware and software systems interoperate with each other. Operating and managing these critical components is not a straightfoward endeavor. To this day, human operators are heavily relied upon for their expert knowledge of these systems and space communication. Much of the heavy lifting in the day-to-day DSN Operations is still done by the operators themselves. With significant advances that have been made in computing over the years, both in hardware (e.g. affordable, faster processors and memory) and software (e.g. automation, rule engines, business intelligence, frameworks for processing big data, high-performance messaging systems, data science, and machine learning), this no longer may have to be the case. By leveraging the combination of these modern, advanced technologies, DSN Operations can be improved on multiple levels: quality of service, additional capabilities, and reduced operation costs.

Figure 2. DSN facilities. Counterclockwise from top left: Goldstone DSCC, Madrid DSCC, Canberra DSCC, and NOCC (JPL).

Taking a cue from other industries, a research and techonology development team at JPL has been exploring the application of *complex event processing (CEP)* in the domain of DSN Operations. CEP is a method of combining streams of data from multiple sources in order to identify meaningful events or patterns. What makes CEP now viable in the domain of DSN Operations are all of the computing advancements mentioned previously. CEP can enable a more comprehensive automation of DSN Operations, reducing the dependency on human intervention. By also using CEP to perform deeper analysis of situations and non-obvious trends, valuable intelligence can be garnered, possibly matching or surpassing a human expert's ability to do so in real-time. Such *operational intelligence* is the DSN equivalent of *business intelligence (BI)* that is increasingly sought after in commercial industries. Furthermore, CEP perfectly fits into the model of the grander Follow-the-Sun Operations (FtSO) that DSN is moving toward. The DSN Operation's CEP system—DSN Complex Event Processing (DCEP) system—provides the very building blocks needed to achieve the FtSO's objective: significant reduction of operations cost.

Currently still in the prototyping stages, the DCEP system has been developed to answer a number of questions:

1) Can CEP really help realize better automation in the DSN?
2) What current problems within DSN Operations does CEP actually solve?
3) What new use cases can CEP handle, that serve as enablers for the FtSO?

These are the questions that this paper addresses in the following sections. In the process of planning and developing the DCEP system, a number of hurdles that exist in the DSN have been identified, and they are distilled in Section II. After a brief description of FtSO (Section III), a more in-depth discussion of what CEP is and how it works is provided in Section IV. Section V presents a set of six real use cases, which the DCEP team used or is planning to use the DCEP system to handle, in order to validate CEP's usefulness in the DSN. Sections VI and VII introduce two open source software that form the core of the current DCEP system; the sections explain why these were selected and what role they each play in the overall DCEP framework. Section VIII revisits the six use cases and describes the CEP solution for each. Section IX provides additional details on how the CEP rules can generally be categorized, and how machine learning can be incorporated to improve CEP. Section X lists some of the work that remains for the DCEP project, and Section XI concludes the paper.

In this paper, when the term "CEP" is used, it refers to the *processing* of complex events, and not the specific system developed to perform CEP. When the term "DCEP system" is mentioned, it refers to the specific CEP system that the authors developed for the DSN (in other words, DSN's CEP system).

II. Current Challenges in DSN Operations

Currently 49 missions, ranging in wide variety including interstellar missions such as Voyager, planetary missions such as Mars Science Laboratory, and Earth satellite missions such as Spitzer Space Telescope, all use the DSN for their spacecraft communication needs. These are not just NASA missions but include those of other space agencies around the world, such as the European Space Agency (ESA), the Japan Aerospace Exploration Agency (JAXA), and the Indian Space Research Organisation (ISRO). The DSN has contributed to the successes of these current missions and numerous other past missions by continuously supplying reliable, high-performing space communication services. In addition to providing communication links to spacecrafts, the DSN itself serves as a valuable scientific instrument. Owing to its catalog of large antennas and precision equipments, it plays a significant role in radio astronomy studies, such as those involving near-Earth objects.

However, the DSN has existed for over 52 years, and throughout those years it has gradually grown in terms of size, capabilities, and complexity. As technology advanced over those years, the computing industry adopted different system and communication standards that came and went in phases. Meanwhile, new subsystems continued to be developed for and delivered to the DSN during that time, and this resulted in modern systems coexisting and interacting with outdated, legacy systems that were introduced years—even decades—before. To address the ever-growing complexity in the DSN and to bring its overall capabilities up-to-date, a number of modernization efforts were undertaken throughout DSN's history. Series of improvements were made to its architecture and infrastructure. However, a complete revamp of the DSN and its comprising elements is simply too costly, and so despite the earnest reengineering efforts, much of the DSN remains outdated and heterogeneous. This situation has held back any improvements beyond what would be considered marginal, or at best incremental, in the DSN, particularly in its operations.

A major goal for the DSN is to reduce the cost required for its operations. The financial budget allocated for the DSN is appropriated annually, and understandably, much of it is allocated to the maintenance and incremental upgrades of its existing infrastructure, which includes many aged equipments and physical structures. A significant portion of the budget is also spent on DSN's day-to-day operations, and this is the area in the DSN where there is continuous concern about costs. The NASA Office of Inspector General recently audited the DSN, and they concluded that reduced budgets for the DSN have resulted in several deficiencies in its operation. This finding also warned that, as a result, some of DSN's future plans may be in jeopardy.[1] Finding ways to cut spending in DSN Operations is now more critical than ever.

The current high cost of DSN Operations is a direct result of DSN's legacy, complex, and heterogeneous infrastructure. DSN Operations still has a heavy reliance on human operators for much of its activities, particularly on the Link Control Operators (LCOs) who manage the spacecraft tracks.[*] There currently is an automation software in use by the DSN Operations, called Temporal Dependency Network (TDN). Actions based on planned sequence of events that the LCOs would normally perform manually are automated using the TDN in scripts. Although very useful, the capabilities and scope of this automation is limited, and a human operator still needs to be dedicated to monitor and control a link (in recent years, up to two links[2]) for actions that the TDN cannot perform. Also, virtually all analysis of anomalies, trends, and patterns need to be performed manually, as the current tools and processes in the DSN Operations provide no automation for those tasks.

If a greater extent of operator and analyst responsibilities can be automated, it would lead to huge cost savings in operations and improved DSN performance. In addition, the groundwork established for such greater level of automation may give rise to opportunities in extracting new insights and faster operational intelligence. However, DSN currently has technical characteristics that presents a significant challenge in introducing such comprehensive automation. Here are some of those characteristics:

A. Heterogeneous Data

Table 1 shows a sample of the various forms of data being produced by their different sources, some of which the human operators in the DSN constantly monitor and inspect.[†] These include real-time streams of monitor data generated by the individual subsystems active in a link and also by the Network Monitor and Control (NMC) software

[*] In DSN Operations, the term "link" is used to denote the logical connection of subsystems and equipments that allow them to interoperate in support of one or more spacecraft tracks.

[†] Not all of these types of data are required for automation of operations, but they are all valuable for analysis and gaining deeper insight into operations.

Table 1. Heterogeneous data in DSN Operations. (Italicized items are those data types that exist in the DSN but have not yet been determined as being required for DSN Operations.)

Data Type	Temporal Nature	Producers	Source for DCEP	Format
Monitor Data	Real-time	DSN subsystems (includes NMC)	MDS (Currently: MON-2/JMS bridge)	Proprietary binary data
NMC Logs (contains operator/TDN directives and responses)	Real-time and historical	NMC subsystem (also TDN)	NMC FileSystem (NMCFS)	Plain text
Other Logs	*Real-time and historical*	*Yet unknown*	*Yet unknown*	*Plain text (most likely)*
Other Real-Time Indicators	*Real-time*	*Yet unknown*	*Yet unknown*	*Yet unknown*
Schedule Items	Planned	SPS	SPS REST server	XML
SOEs	Planned and predicted	SPS	SPS REST server	XML
Astrophysics Data	Predicted	LTPS	LTPS REST server	Plain text
SQA Data	Historical	DSN subsystems (includes NMC)	SQA server	PL/SQL collections
DRs and MDRs	Historical	Operators	DRMS	Form data in English

itself.[*] These monitor data flow through the Monitor Data Service (MDS), which is part of the Monitor and Control Infrastructure Services (MCIS). Monitor data is exchanged in a publish-subscribe fashion, and accessing this data requires the use of the NASA-proprietary MCIS library, which is based in the C language. Monitor data provides continuous indication of the health status—as well as some activity and configuration states—of the different subsystems.

Logs are another important category of data. NMC, TDN, subsystems, and others generate activity and system logs that provide valuable information about both the real-time events that are occurring and also what have happened in the past. The LCOs, using a graphical user interface (GUI) running on their computer workstations, issue directives to NMC and subsystems in order to have them perform certain actions (e.g. lock on the signal) or change their configuration states (e.g. changing the bit rate). Both these directives and the resulting responses are recorded and retained in the NMC logs. Also, TDN-generated log entries (included as part of NMC logs) give indication of its automation status, and the LCOs will monitor this in case they need to intervene and resolve issues.

The Service Preparation Subsystem (SPS) produces Schedule Items, and each contains information about a spacecraft's track, such as start and end times, the antenna used, and to which higher-level support activity it belongs to. These Schedule Items are produced well in advance of the actual track, to support the operation's planning activities. SPS also generates the Sequence of Events (SOE) files, which lists in time order the configuration and actions that subsystems need to take as well as the external events that are predicted to occur (e.g. Mars occultation). SOEs are provided in both human-readable text file format and—more recently—in XML. Both of these data types are available for the future, but they are also archived. This also makes them useful for post-analysis. TDN relies on SOEs and the aforementioned real-time monitor data for its automated execution of operator actions.

Another type of useful data is produced by the Light Time Physics Service (LTPS), which provides highly accurate and precise astrophysics data in real-time. For example, a LTPS query can return the value of the one-way light time between a DSN antenna and the spacecraft of interest, such as Voyager 2, at a specific moment in time. This information can provide greater context during analysis: for example, helping to clarify the source of downlink signal and noise level fluctuations observed during a track.

Also of important value and interest is the Service Quality Assessment (SQA) subsystem's repository of historical monitor data. The real-time monitor data mentioned at the outset of this subsection is completely transient. SQA captures a key subset of the monitor data that and stores them in a database. The monitor data archived in this persistent

[*] NMC is a subsystem used to configure, control, and monitor other subsystems. It serves as the main conduit for LCOs.

data store ranges in time from 24 hours to several years in the past. This information repository can be valuable for comparing patterns observed at the present time to snapshots in the past.

The final type of data that needs mention is the reports filed by the operators themselves, called Discrepancy Reports (DRs). Anytime there is a failure to support a scheduled DSN activity or an interruption during a support, the operator writes a DR and enters it into an online repository called the Discrepancy Reporting Management System (DRMS). If there is a recurring problem, rather than file a separate DR with each incidence, a single Master DR (MDR) is filed. Previously filed DRs and MDRs are useful when assessing a newly encountered problem because they inform the operator whether or not the same problem has occurred in the past and what corrective actions were taken. Currently, trying to accurately match a new problem with any of the problem signatures captured in existing DRs and MDRs involves much manual analysis.

All of these different types of data, in their own unique formats and properties, are only available from disparate sources. An intelligent system that can provide truly extensible automation and analytical capabilities must consume all of them, and this is not a trivial objective to accomplish.

B. Inconsistent Naming Conventions

As mentioned earlier, over the course of time new subsystems have been developed and then integrated into the DSN in stages, and the newer subsystems did not always stick to the conventions and interfaces of their legacy predecessors. An area where this presents a problem is the lack of *standard naming*, particularly with monitor data. For example, the Antenna Control Assembly (ACA) instances provide the monitor data of the 70-meter antenna in Madrid and Canberra and the 34-meter high-efficiency (HEF) antennas. ACA publishes the current azimuth value of its antenna using the character string "AZANG" and the elevation as "ELANG". On the other hand, the Antenna Pointing & Control Assembly (APCA) instances, which provide the monitor data of the 70-meter antenna in Goldstone and the 34-meter beam waveguide (BWG) antennas, publishes the same pair of values under different character strings: "AzimuthAngle" and "ElevationAngle". This is merely one example of the yet undetermined number of cases in the DSN where subsystems do not share a common naming convention.

This situation causes additional complexity for automation and analysis, as additional associations need to be declared. In the example above, the system must somehow *know* that "AZANG" and "AzimuthAngle" monitor data are not distinct data items, but instead, that any computation performed on one can equally be done on the other.

C. Distinct Attributes Among Same Subsystem Types

The same class of subsystems may exhibit different properties, and these inherent uniquenesses also present challenges. Even among a single class of subsystems, one instance may produce data that have very different values than another instance, although both of them are effectively in the same state and the data indicates that. Using again the antenna pointing assemblies as an example, an assembly for an antenna at Canberra will generate "AzimuthAngle" and "ElevationAngle" pair of monitor data as 0.10 and 90.07, respectively, when the antenna is in the stow position. The assembly for another antenna in the same complex, however, will generate the pointing angle values of 45.00 and 89.00 when stowed. These types of attribute discrepancies among seemingly identical subsystems serve to add another layer of complexity.

D. Non-Scaling Monitor Data Service

MDS, as well as its parent MCIS, were created in the 1990s to provide a common monitor and control communication infrastructure for the operators and the DSN subsystems. It had a robust design, with redundancies and failover features built in, and even to this day continues to provide its services reliably. In the years that followed, however, more and more MCIS client software were produced, and it became apparent that MCIS, particularly its MDS server software, is prone to suffer degraded performance and even failures if there are large number of clients taxing its services at the same time. This is an issue because we cannot freely introduce new software that subscribes to monitor data to the MCIS ecosystem, for the fear that DSN's support of missions will be adversely affected, even causing complete outages. Furthermore, the MCIS client library also has its limitations, where it simply cannot handle large numbers of monitor data subscriptions. The subsystems in the DSN, as a collection, produce thousands of monitor data items at very high rates. A standard MCIS client is unable to subscribe to all of these items and not have its memory usage grow unbounded.

Changing the MCIS design or even replacing it entirely with one that can scale to meet the growing demand on its services is deemed too costly—both in terms of finance and risk. So in order to augment the DSN to enable better automation and operational intelligence—which will definitely need access to monitor data in real-time—while maintaining the same level of service quality for the missions that it supports, the infrastructural handicap that exists in MCIS needs to be overcome.

E. Restricted Wide Area Network Bandwidth

Another limitation in the current DSN infrastructure is in the network's physical layer: Its wide area network (WAN) has a restricted bandwidth. This severely curbs many potential capabilities in the DSN, particularly remote operations and DSN-wide analytics. Much raw data generated at the DSCCs need to be downsampled, or perhaps curated, before they are transmitted over the WAN. This, in fact, is the case: DSN replicates in real-time just a couple of hundred monitor data items from the DSCCs to the NOCC, and the rate is downsampled. This reality is less than ideal, because products we can extract from analysis and operational intelligence can only be good as the data being collected.

The sample of technical issues just discussed leads to a number of undesirable realities in the current DSN Operations. For one, automation becomes non-trivial. Also, analyzing situations and making decisions is slowed due to the fact that much of the process is manual. There are anomalies and deviations from the norm that go undetected because there is not a system or process that will take into account all the metrics available. Similarly, opportunities are being missed for recognizing patterns and trends that can provide knowledge that contribute to improving the operations. Because a system that can provide fast, intelligent notifications and recommendations to the human operators does not yet exist, the operators continue to monitor the subsystems and make all the decisions without any help, except for the small-scale automation that TDN currently provides. In the big picture, these all result in continuous heavy reliance on human operators, leading to the high cost of DSN Operations.

III. Follow-the-Sun Operations

JPL has undertaken a project called Follow-the-Sun Operations (FtSO) to improve DSN operation's cost-efficiency in the upcoming years.[2] FtSO involves a pair of shifts from the current operational paradigm:

A. Remote Operations (RO)

Presently, LCOs are staffed at each DSCC twenty-four hours a day, seven days a week. There are three working shifts per day, at every DSCC, round-the-clock. FtSO aims to reduce this staffing load by making use of remote operations. In the new paradigm, each DSCC will have only one working shift per day during daytime, and LCOs on duty will not only operate the local DSCC assets but also remotely operate assets in the two other DSCCs. The term "follow-the-Sun" derives from this strategy, that all DSN Operations are handled at the DSCC where the sun is visible in the sky.

B. Three Links per Operator (3LPO)

As mentioned in the previous section, each LCO currently manages up to two links at a time. In FtSO, each operator will handle up to three links. This will reduce the required human staffing even further.

FtSO is promising in that the efficient staffing of human operators will achieve significant savings in operation costs, while continuing to provide high-quality service to DSN users. In order to realize FtSO, however, the technical difficulties explained in the previous section need to be addressed. Feasibility of RO is contingent upon larger amounts of information being exchanged between the DSCCs over the WAN than what is currently taking place. If the switch to 3LPO happens without reducing the LCOs' work load per link—by at least one-third, theoretically—it will lead to overburdening the staff. To solve these problems, the research and technology development task DCEP is investigating these challenges as part of the FtSO project.

IV. Complex Event Processing

CEP, as a technical term, is a method of taking as inputs a variety of real-time data from different sources, and then combining them to determine their significance and to take action.[3] The data may be of different types, and they may contain information that are seemingly unrelated. A CEP system transforms the data as necessary, and then analyzes them to detect relationships, trends and patterns. This continuous process often results in real-time actions, in addition to producing artifacts for analysis that provide deeper insight into the collective data.

Many industries have been using CEP successfully for years: financial markets, retail, homeland security, intelligence agencies, social applications, et cetera. As an example, when a person's credit card is swiped at a sales register in Pasadena, California, but then the same card is swiped again at a register 2220 kilometers (1379 miles) away in Houston, Texas, merely five minutes later, the second transaction is automatically denied. This is because the CEP system at the credit card company quickly processed the credit card information of these two transactions, the times and geographical locations at which these transactions were made, and also probably the historical usage pattern

of the credit card, then it determined that there likely is a fraudulent activity among the two transactions that just took place. The different pieces of information involved in this analysis may have been collected from very different data sources. This is an example of how CEP *detects* meaningful events.

CEP is also useful for *predicting* meaningful events that are likely to occur. In the stock market, for example, many of its traders depend on algorithmic and high-frequency trading systems to maximize their profits. The success of such trading systems is largely driven by their ability to simultaneously process large volumes of information, some of which that are discrete in nature. To illustrate, a CEP system in this domain may not only process real-time streams of global stock prices and foreign exchange rates but also news headlines. News of company mergers and wars in countries are just some of the data that can drive prices up or down in markets. A CEP system with the intelligence to accurately predict what events will likely follow (e.g. plummeting stock prices) will produce actions to make good use of opportunities and to avoid disasters (e.g. unload the entire portfolio of stocks that are about to lose value).

The real advantage of CEP systems over other types of systems such as business rules engines is its ability to correlate data, thereby framing data into contexts. To illustrate its importance, we can use as an example a real-life incident that occurred in 2012. A young Irishman posted a message on Twitter, saying: *"Free this week, for quick gossip/prep before I go and destroy America."*[4] Few weeks later when he arrived at the Los Angeles International Airport, he was taken into custody by the Department of Homeland Security agents. The United States government constantly scans social networking services for information that may suggest upcoming acts of terror, and in this case, the young man's tweet had become a serious item of interest because of his use of certain key words ("destroy America"). After interrogation, however, it was clear that the young man had no intention of committing terrorist acts. The word "destroy" is a common British slang for partying and getting drunk, similar to the American slang "(getting) trashed." Had the word-watch system taken into account—or put into context—that the originator of the message is from the British Isles and in that region the word "destroy" may be used as a slang to mean something that is not so threatening, the traveler could have been spared from his ordeal. Although this particular incident did not result in any permanent or serious damages, there are situations where intelligent association of information to form proper contexts can either lead to important consequences or avoid disastrous ones. CEP systems can be used for processing, recognizing, and correlating such different types of relevant information, as well as for supporting temporal correlation of events.

Many industries such as financial, retail, and social media have leveraged CEP to reduce their operating costs, while at the same time adding improved services and capabilities that simply would not have been possible without it. More recently, these enterprise industries seem to be investing in two areas to improve their operations: *business intelligence (BI)* and *data science.* According to Gartner, an information technology research company, BI "is an umbrella term that includes the applications, infrastructure and tools, and best practices that enable access to and analysis of information to improve and optimize decisions and performance."[*] Data science is a relatively new, multidisciplinary field that is focused on extracting knowledge and insights from data in various forms. As many enterprises faced the challenge of extracting value out of their *big data*,[†] this naturally ushered in data science. By many definitions, DSN also is an enterprise, and using CEP as the enabling technology, BI and data science methods can equally be applied to DSN Operations much to its benefit.

To begin with, CEP can provide solutions to the two main problems faced by the FtSO project, namely the bandwidth restrictions in the DSN's WAN and the heavy workload imposed on the LCOs under the 3LPO scheme. Rather than transmitting the entire large scale of data produced at a controlled DSCC to the controlling DSCC (RO complex), the CEP system running at the controlled DSCC can process all the data onsite, and then transmit only the events or data items of interest to the RO complex. The objective is to 'find signal from the noise' of endlessly generated volumes of data and use the WAN resources only for these 'signals'. This entails the CEP system processing all of the heterogeneous data available at the DSCC, from real-time monitor data to SOEs to DRMS records and correlating them. (See Figure 3.) Meanwhile, as the CEP takes over more of the data analysis and forensics on behalf of human operators, the amount of attention and routine actions required by the LCOs can be reduced. As automation is pushed towards a more lights-out process, LCO's workload can be managed to reasonable levels. To that end, the following section discusses some of the specific use cases that the CEP system in the DSN Operations domain—the DCEP system—needs to handle.

[*] http://www.gartner.com/it-glossary/business-intelligence-bi/

[†] "Big data" can be defined as extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions.

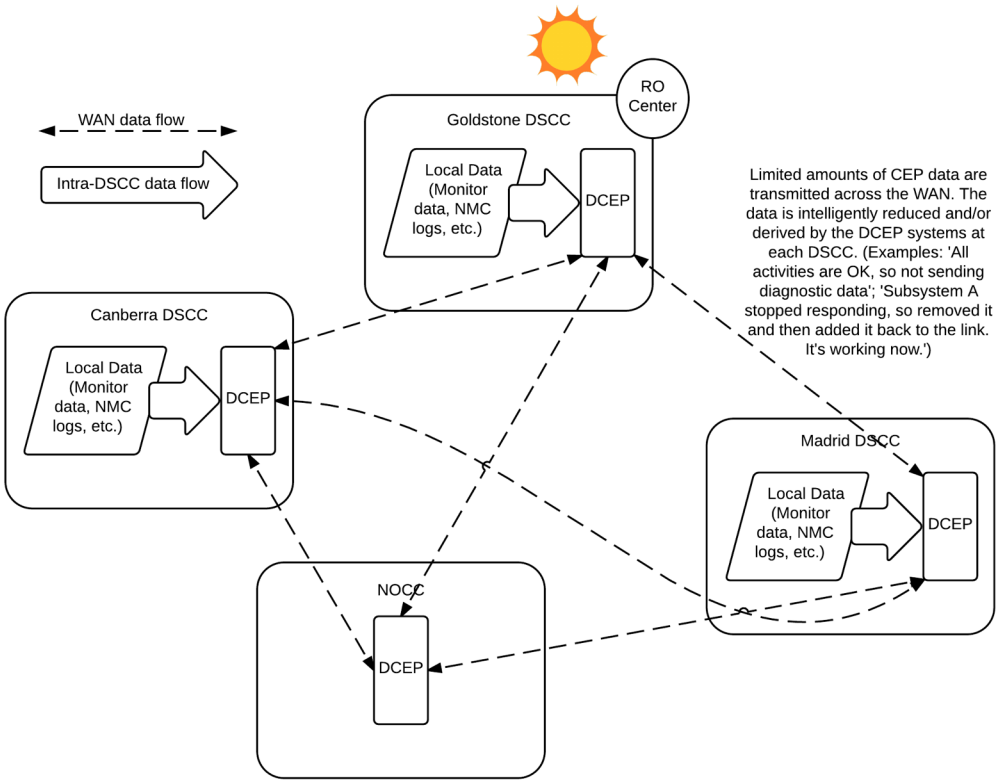

Figure 3. Using DCEP as the building blocks for FtSO. The DCEP systems running at each DSCC process the voluminous, locally-generated data. After performing CEP on the data, only meaningful information is transmitted across the WAN. For instance, when everything is going well at the remote DSCCs, the DSCC serving as the RO Center does not need to receive much data—only periodic notifications that indicate all services are healthy.

V. Use Cases

Many use cases have been identified for which CEP can make the case for its usefulness by providing novel solutions, as these use cases presently pose challenges to both the current DSN Operations as well as to the future FtSO. In this section, six of them are introduced.

A. Standard Naming

As discussed in a previous section, DSN data currently falls short of having a uniform, standard naming scheme, and this complicates automation and analysis. When an analyst queries for the azimuth and elevation values for all the antennas, for example, the analyst should not have to worry about whether he/she has queried for them all or if there is needed data that falls outside the query because of any naming differences. Someone who has many years of experience in the DSN and has extensive knowledge of it may not consider this too big of a hassle, but this sort of intricacy is what keeps the operating of the DSN heavily dependent on the domain experts. CEP should, therefore, handle these naming complexities rather than leaving it to the operators and analysts to figure out and determine.

B. Framing Events in Context

The DCEP system should go above and beyond simple limit checking when alerting (or not alerting) the operators about a situation. For example, when the downlink signal from the spacecraft is lost in the middle of a track, this can indicate a fault and operator intervention may be required. However, it is also possible that the spacecraft simply has entered a planetary, solar, or lunar occultation, in which direct visibility is lost. In this case, the loss of signal is not a fault but rather a normal event. The LCO need not be needlessly alerted by this event, since there is no action that the LCO can take to recapture the signal, and at the end of the occultation the downlink communication should resume

by itself. To put it humorously, the LCO can continue to have his coffee. Even if it is desirable that the operator be alerted, he/she should be provided with the additional contextual information that the loss of signal was due to an occultation and therefore is not a deviation in the DSN service.

It is worthwhile to consider the converse situation. If the spacecraft is in an occultation but the downlink subsystem reports that it is receiving signal (possibly due to a malfunction), this would be an anomalous situation and the operator may want to be alerted when this occurs, or at least have this incidence automatically recorded so that an investigation can be performed later. With systems that perform only simple condition or limit checking, this sort of anomaly would go undetected unless the LCO himself is monitoring the situation closely.

C. Detecting Deviations from the Norm

Some deviations happen so gradually over a long period of time that operations may not be aware of it until there is a total failure. To illustrate, Deep Space Station (DSS) 43 at the Canberra (Australia) DSCC regularly tracks the Voyager 2 spacecraft. Suppose that the operator notices the signal-to-noise ratio (P_c/N_0) measured by the downlink subsystem to be 0.1 units lower than the day before. Nonetheless, the operator is not concerned because the signal is in lock and data is being received. Few days later, during another support for Voyager 2 (again on DSS 43 and at around the same time of day), the operator notices that the P_c/N_0 is back up by 0.1 units. So it all seems that things are very stable. However, compared to the year before, the average P_c/N_0 during those few days may actually be off by 10 units. There can be any number of reasons for this, such as failing hardware or loss of calibration. The takeaway is that differences in these kinds of measurements may take place so gradually that unless statistical comparisons are made between the current and the historical, some deviations will never be caught until visible failures occur. Detecting these deviations early, thereby avoiding the after-the-fact investigation work and downtimes, would mean higher quality of service to DSN customers.

D. Matching Incidents to Known Discrepancies

During operations, when there is a failure or interruption of service, one of the things that greatly assist problem resolution is finding out whether or not the encountered anomaly has occurred in the past and how it was resolved. As mentioned previously, in DSN Operations, these issues are recorded as DRs. If a particular discrepancy is a recurring problem, it is instead documented as a MDR. So, for example, if a problem were to arise and the incident is quickly matched to an existing MDR, that alone would save the operator from the work of filing a new DR and also the work required to search through past DRs and MDRs to see if it is was a reoccurrence. This would be one of the most basic benefits that DSN Operations can gain from this use case. Furthermore, the ability to match the new problem to an existing DR or MDR in real-time will most likely provide the operator with useful knowledge of how to address the problem, without his having to analyze and do problem-solving from scratch.

E. Generic Tools

Currently there is no way to quickly create an application or write a script that will easily interface with the different data sources that exist in the DSN. Being able to do so will be extremely valuable, however, because over time new service requirements are placed on the DSN and providing new capabilities can be expensive without such a framework. For instance, in 2014, the JPL Executive Policy Committee asked that the DSN track the spacecraft command-loss timer for each mission as part of the NASA Continuity of Operations Plan (COOP). Many DSN mission spacecrafts carry a timer onboard: If a command from Earth is not received before the timer expires, the spacecraft will then enter safe-mode. The new requirement has been placed on the DSN in order to avoid those situations. To track when each of the spacecrafts will time out due to not receiving a command, the option that will provide the best accuracy is to process the command counter data contained in the downlinked spacecraft telemetry. However, there is no existing interface for the DSN to access each missions' spacecraft telemetry in real-time. There is another option: keep track of the last CLTU radiation time for each spacecraft. At best this would help estimate the timeout values, since a spacecraft will not receive the command bits until the delay of one-way light time (OWLT) has passed, and also there is no guarantee that the radiated data will be captured by the spacecraft. However, both types of data that are needed for this estimation already exists inside the DSN: real-time monitor data that indicates when the last CLTU was radiated and the table of timeout values for each spacecraft. In order to produce an automated tool that will process these data and track the spacecraft command-loss timers, one would presently have to create a software application that uses the MCIS library, and also undergo scrutinizing reviews and testing to ensure that the existing MCIS services will not adversely be affected by the introduction of this tool into the real-time DSN Operations. Also, because of the dependency on the MCIS library, the software may need to be written in C or C++, severely limiting the implementation options. Ideally, analysts, engineers, and operators should be able to write custom tools quickly and easily, that can process DSN data without the heavy overhead and inflexibilities in implementation.

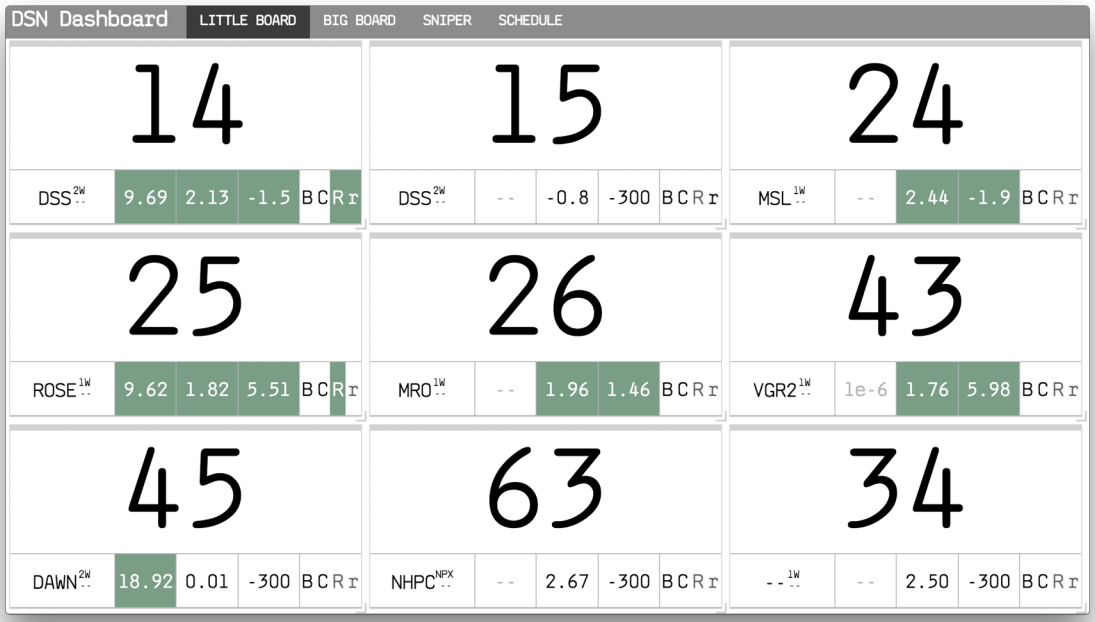

Figure 4. Sample of the Postage Stamp display. Large numbers indicate the DSSes, and the smaller icons display the different states of the links. Icons change colors depending on whether things are going as expected (green) or not so much. (Image credit: Dr. Alexandra Holloway)

F. Postage Stamp

As part of one of the latest DSN Operations modernization effort, the Human Interfaces for Mission Operations group at JPL is conducting a parallel study on a new user experience design for the LCOs. This new design would provide real-time link situation information to the LCOs in a more intelligent and user-friendly way than what is currently being provided. Figure 4 shows a sample of this new design. Because of the way the GUI elements are laid out on the display, the name given to this new design is Postage Stamp. In order to provide this user experience, the Postage Stamp system needs to process real-time data and produce higher-level, correlated information. The required real-time data include information on downlink and uplink ranging, commanding, symbol status, stage of track, predicts mode, antenna state, et cetera. In the existing DSN infrastructure, the Postage Stamp team would need to create an MCIS client software to subscribe to the various monitor data that carry the aforementioned real-time information, write their custom algorithms to derive the desired higher-level, correlated information out of those data, and then have the resulting data displayed on the GUI. However, similar to the previous use case, this entails much development and testing work. It would be far more desirable for the Postage Stamp system to access the data in a much easier way, so that the Postage Stamp design team can focus on their primary work which is user experience.

So far, a number of use cases that CEP can address to improve upon the existing capabilities of the DSN have been described. Successfully handling these use cases will bring much benefit to DSN Operations. In the following sections, the approach that the DCEP team has taken to handle the use cases, with its design and implementation, will be discussed.

VI. Fast Data Processing Using Apache Spark

A CEP system that will process the large volumes of real-time, historic, and predicted (commonly referred to as "predicts") data in the DSN needs to be high-performing and scalable. As a result of many years of development and experience in this field by the enterprise industries, several viable products—most of which are commercial—that provide CEP are available in the market: TIBCO's StreamBase, EsperTech's Esper, SAP's Event Stream Processor, Red Hat's Drools, to name a few. Products such as Splunk market themselves as a "platform for Operational Intelligence." The DCEP team evaluated some of these commercial off-the-shelf (COTS) products. As with many

COTS products, these solutions have a number of significant disadvantages: high licensing costs, vendor lock-in, and lack of freedom due to the software being closed-source. It is unclear whether or not the DSN operation costs can ultimately be reduced despite these factors, but even just the investment in research and prototyping work using these COTS solutions alone would have incurred significant amount of cost. The DCEP team therefore eventually focused attention on the free and open source software solutions.

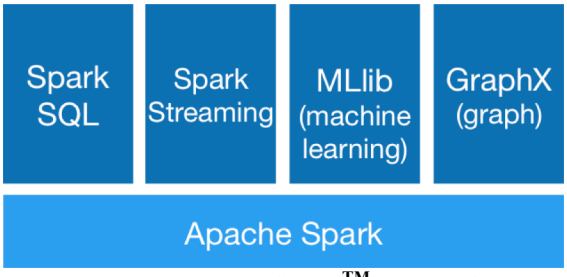

Figure 5. The Apache Spark™ stack of engine (bottom) and libraries (top). (Image credit: http://spark.apache.org/)

With the advent of big data challenges in enterprise industries and other fields, a number of open source software platforms that can perform large-scale parallel and distributed data processing emerged. At present, one of the most popular of such platforms is Apache Spark™. Apache Spark is "a fast and general engine for large-scale data processing."* Unlike Hadoop MapReduce, which is another big data processing engine, Spark stores data in-memory rather than exclusively on disk. This allows Spark to achieve processing speeds that are ten to hundred times faster than Hadoop MapReduce.[5] Using Spark, it is possible to process and combine large volumes of data from multiple sources fast, and so it can serve as a solid foundation for complex event *processing*.

Apache Spark has a modular design where there is a core engine (Spark Core) and working on top of that engine are four built-in libraries: SQL, Streaming, MLlib and GraphX. Figure 5 is a visualization of this design. Particularly useful to the needs of CEP are the SQL and Streaming libraries (in future work, MLlib also). Spark SQL allows Spark applications to query structured data using SQL, and these queries can be uniform regardless of the data source. To show a simple demonstration on how this works, Figure 6 shows a sample Spark application code that uses Spark SQL to extract data of interest. This particular code is written in the Scala language, although there are other language choices, such as Java, Python, and R, for writing Spark applications. The example code loads the latest real-time monitor data captured (saved as a JSON object in an Amazon GovCloud S3 bucket, just to demonstrate) and also queries the historical monitor data repository in the SQA subsystem. The objective is to visually see if the P_c/N_0 just observed for the Cassini spacecraft is same or different than the historical average. Because both data are structured, we can use simple SQL queries to extract the data. At the same time, we can use Spark's MapReduce programming pattern to create a flow of data transformations and actions. In Spark, no work is actually executed on *transformations* of data but rather only on *actions*.† Multiple transformations therefore can be chained together without triggering any processing of the data. This lazy evaluation model is preferential when dealing with large streaming data sets, as it ensures that only those computations directly related to the chosen *action* are executed—others are ignored.

```
val sc: SparkContext // An existing SparkContext.
val sqlContext = new org.apache.spark.sql.SQLContext(sc)

val monDataDF = sqlContext.read.format("json").load(
  "s3n://bucket/latest-mon-data.json").filter(monDataDF("mdItemName") ==
  "receiver.pcno").filter(monDataDF("mission" ==
  "CAS").registerTempTable("latestCASpcno")

val historicalDF = sqlContext.read.format("jdbc").options(
  Map("url" -> "jdbc:oracle:thin:user/password@//sqahost:1521/archivedmondatadb",
  "dbtable" -> "historydb")).load().filter(historicalDF("mdItemName") ==
  "receiver.pcno").filter(historicalDF("mission" ==
  "CAS").registerTempTable("historicalCASpcno")

sqlContext.sql("SELECT mdItemName, value FROM latestCASpcno").show
sqlContext.sql("SELECT mdItemName, AVG(value) FROM historicalCASpcno WHERE timestamp " +
  "like '2015%'").show
```

Figure 6. Sample Spark application code in Scala.

* http://spark.apache.org/
† This is called *lazy transformations.*

Figure 7. Spark's Streaming library splits the incoming data flow into data batches of small sizes. These in turn can be processed just like any other data set in Spark. (Image credit: http://spark.apache.org/docs/latest/streaming-programming-guide.html)

The Spark Streaming library allows Spark applications to consume continual streams of data in a scalable, high-throughput, and fault-tolerant way. It provides a number of (stream) receivers out of the box for TCP sockets and popular data streaming systems: Apache Kafka, Apache Flume, Twitter, ZeroMQ, and Amazon Kinesis. Custom receivers can be implemented (say, for example, one that subscribes directly to MCIS data). Same transformations and actions available in Spark Core can be performed on data received by Streaming. Figure 7 shows how Streaming divides an incoming data stream into small batches, which then can be processed as any other data in Spark. However, there are problems when the streaming data is blindly divided into microbatches: Two or more data items that should be computed together may end up being split apart (e.g. the azimuth value in batch N and the elevation value in batch N+1). With this in mind, the Streaming library provides *windowed computations,* which allow applications to apply transformations over a sliding window of data whose size can be larger than that of the batch itself. The library also offers configurable persistence levels, such as replicating the input stream of data into multiple nodes in order to provide fault-tolerance (i.e. no streaming data is lost even when one or more nodes fail).

The DCEP team currently runs a Spark cluster of thirty-two CPU cores on different virtual machines in order to prototype and demonstrate the CEP use cases. (See Figure 8.) A set of perpetual Spark jobs are run to process the input stream of real-time monitor data from the DSCCs. As part of the processing, these real-time data are compared against the planned operational events (SOEs of the spacecraft tracks) to check if the DSN support is proceeding satisfactorily. Also, another job continuously correlates and translates the real-time monitor data to produce the JSON data consumed by the Postage Stamp system. Much of these operations include SQL queries on the streaming data, and all of the jobs are programmed as Spark applications in either Scala or Java. Further discussion on how this Apache Spark-based framework is being used to handle the different use cases introduced in Section V will be included in a later section.

The CEP Spark applications need access to different data sources in order to handle the DSN Operations use cases. Some of the data types listed in Table 1 have RESTful APIs[*]: Schedule Items, SOEs, and Astrophysics Data. The nature of these data types is also "on-demand"—the CEP Spark applications only need to pull or fetch the data as needed, rather than consuming them as streams. Sizes of these data are small, in the order of kilobytes, and so it is fairly quick to retrieve them. Using REST's GET operation directly in the application code is therefore suitable. Accessing the archived historical data in the SQA subsystem, on the other hand, is not as straightforward. The repository contains last nine years of collected DSN monitor data, and its size is in the order of tens of terabytes—which will only grow

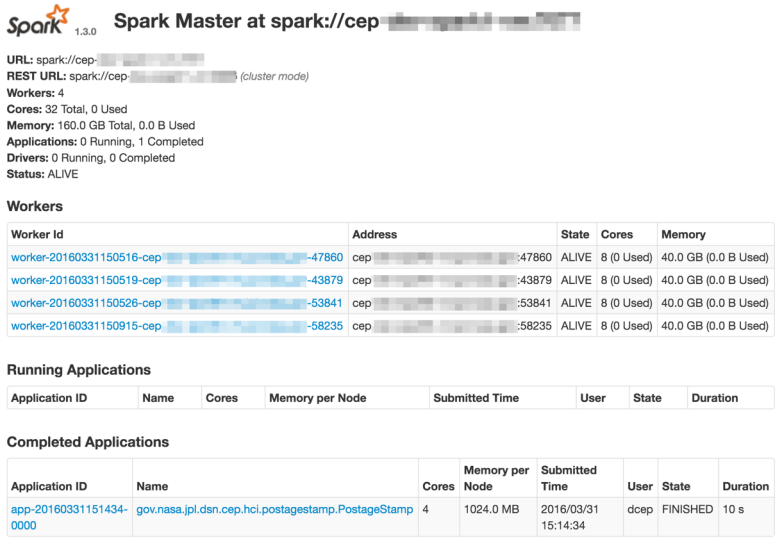

Figure 8. Web user interface of the Spark cluster's master node.

[*] REST stands for representational state transfer. If a system or an interface conforms to the REST constraints and supports its operations, it is said to be RESTful. API stands for application programming interface.

over time. The current API design for accessing this data involves use of PL/SQL functions[*], with a set of historical metrics pre-calculated by the SQA. This strategy helps ensure that data retrievals take a reasonable amount of time and that the SQA subsystem's resources are not overburdened (which would happen if the DCEP system directly submits SQL queries on non-indexed data via JDBC[†]). At this time, there is no interface for Spark applications to access the DRs stored in DRMS. Because DRs are written by the operators in human language, they require a measure of extract, transform, and load (or ETL, a common process in data warehousing) before they can be retrieved and processed in a programmable way. Alternatively, machine learning may be used to have the DCEP system automatically interpret the original DRs. This approach will be explored in future work. In the meantime, use cases that involve DRs are handled by implementing the discrepancy conditions directly inside Spark applications.

The most critical data source for automating operations in the DSN is the real-time monitor data that are continuously generated by the DSN subsystems. For receiving the input stream of this data, rather than implementing a custom receiver in Spark that will tap directly into the MCIS infrastructure, the DCEP team implemented a more modular data streaming configuration using an existing MON-2[‡] to Java Message Service (JMS) bridge and the Apache Kafka messaging system. In addition to monitor data, the DCEP system also leverages this messaging configuration to also consume real-time logs, such as the NMC (including TDN) logs. The following section focuses on this input data stream solution using Apache Kafka.

VII. Scalable Messaging Using Apache Kafka

Although Apache Spark allows for creating custom receivers for input stream data, the DCEP team opted for an alternative solution. When evaluating and prototyping with different processing engines, a more cost-effective approach is to modularize the data input stream separate from the processing system, so that different processing applications can be swapped in and out without having to write custom code for the input adapters. So rather than implementing a custom receiver in Spark that directly subscribes to the MCIS monitor data—DCEP system's main real-time data type, the DCEP team created a "plumbing" software that receives the monitor data and pushes it out to a message bus. This is beneficial in at least two ways: (1) Because the monitor data is published to a common-access message bus, as long as the message bus is scalable, any number of clients can consume the data without adversely impacting the existing DSN service; (2) subscribing to the DSN monitor data actually requires much complicated logic, so it is better to abstract this away from the processing engines under evaluation. (DSN monitor data subscription contexts are dynamic and frequently change according to the current link configurations, so keeping track of these link states and performing unsubscribe-subscribe when these states change is imperative for assuring continual flow of data.) In this data streaming configuration, the DCEP team chose Apache Kafka as the message bus.

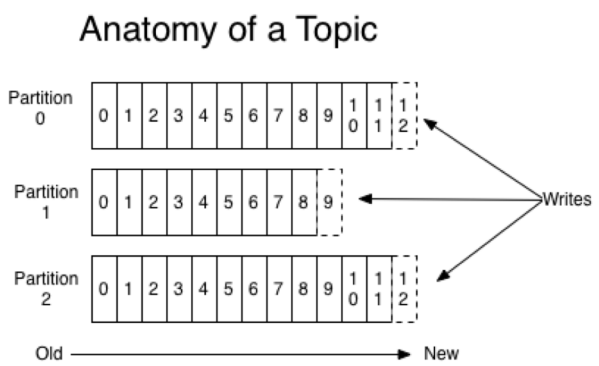

Figure 9. In Kafka, a message topic is nothing more than a partitioned log. Each appended message is assigned an incrementing offset. (Image credit: http://kafka.apache.org/documentation.html)

Apache Kafka is a distributed publish-subscribe messaging system that works well with Apache Spark. Kafka shares many features in common with other message queueing systems, such as using topics as the primary abstraction for publishing and subscribing to related data. However, Kafka has some unique features that make it more advantageous over traditional messaging systems. One of those features is Kafka's treating each topic partition as a *log*. A new message placed into the partition is assigned an incrementing offset and is simply appended to the end of the "log." (See Figure 9.) The responsibility for ensuring reliable message delivery is on the

[*] PL/SQL stands for Procedural Language/Structured Query Language. It was developed by the Oracle Corporation to allow procedural operations to be performed on their databases, extending the capabilities afforded by the standard SQL alone.

[†] JDBC stands for Java Database Connectivity. It is an API for the Java programming language. Java applications use this API to connect to databases and access their data.

[‡] Strictly speaking, MON-2 refers to the DSN Monitor and Control Standard. At times, however, the term is used to refer to the monitor data transport protocol, as is the case here.

consumer: The consuming client needs to track the offset of the last message it receives. If the consuming client's input stream is interrupted (or the client fails entirely), upon recovery, the consumer needs to provide Kafka the offset value from which it wants to resume receiving data. This design allows consumers to come and go without much impact on the Kafka cluster or other consumers. Coupled with other features such as replicable partitions and allowing no more than one consumer group per partition, Kafka also takes advantage of fast sequential disk access which reduces I/O overhead, resulting in high throughput.* Apache Kafka, like Apache Spark, is being widely embraced by industry for scalable data processing tasks.

The plumbing software that does the actual subscribing to the DSN monitor data takes advantage of an existing MON-2 infrastructure between the DSCCs and JPL. This infrastructure converts a limited—but most important—set of monitor data from the DSCCs into JMS messages and makes it available for other software applications in the JPL network to subscribe to. As a precautionary measure, in order to not impact existing services that already make use of the MON-2/JMS bridge, the DCEP team uses a replicated JMS broker (i.e. repeater of the main broker) and its own MON-2/JMS bridge. This ensures that the DCEP team's work does not have any impact on both the operational DSN services and other JPL services that rely on the bridged monitor data. Figure 10 shows the overall architecture of the DCEP system, its data sources, and the Kafka message bus.

VIII. Operational Intelligence

Using the CEP solution described thus far, each of the use cases that pave the way for fast and intelligent DSN Operations introduced in Section V can now be handled. This section revisits those individual use cases and presents the CEP solution for each.

A. Standard Naming

Addressing the lack of standard naming of data should be considered as a prerequisite for automating DSN Operations because handling of other use cases can be made simpler as a result. Using the current CEP solution, standard naming of the real-time monitor data is achieved by executing a name translation job in Apache Spark. This involves two topics in the common Apache Kafka message bus: a raw input topic and a standardized name output topic. Figure 12 shows how this flow of data and name standardization works. Currently the plumbing software adds

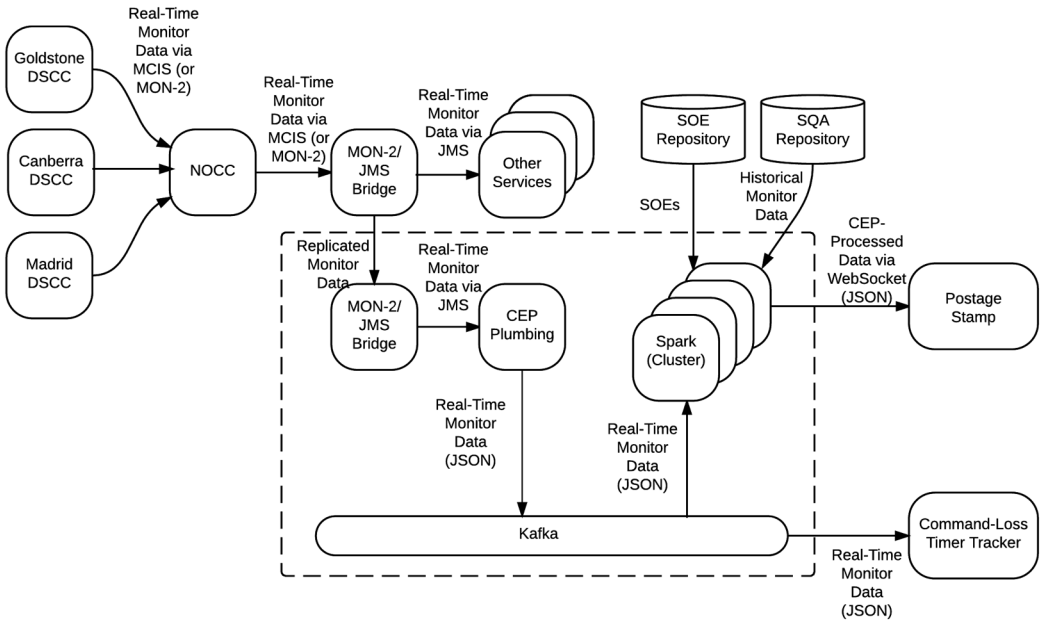

Figure 10. Current architecture of the DCEP system. The dashed boundary line shows what is included in the scope of the DCEP. NMC logs and DR/MDRs will be integrated in the future as additional input data sources.

* http://kafka.apache.org/documentation.html#maximizingefficiency

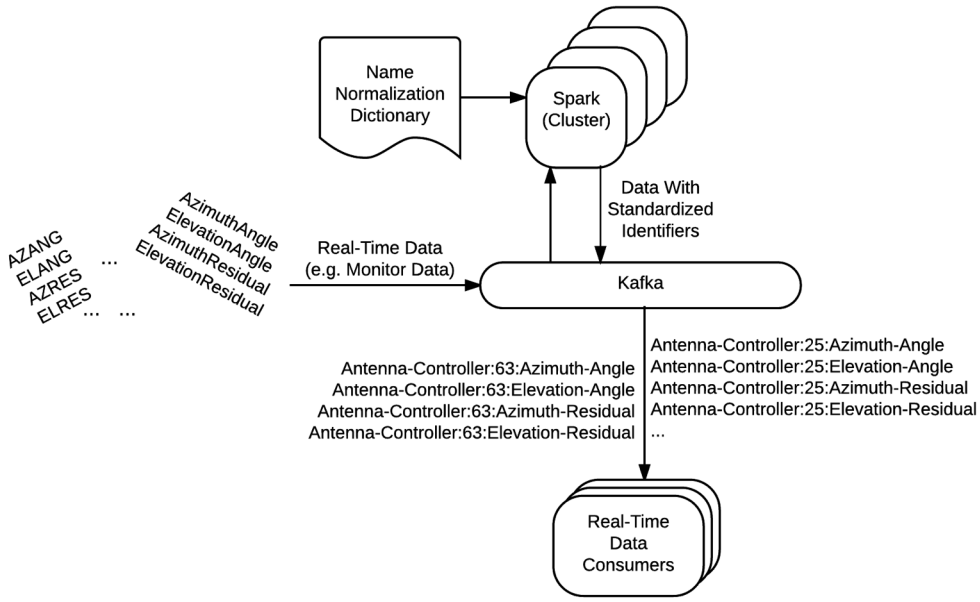

Figure 11. DCEP's name standardization Spark application(s) translates real-time data with nonuniform identifiers into those trivially recognized by the consumers.

additional metadata to each sample of monitor data to make it more meaningful, such as the originating subsystem, timestamp, unit of measurement, a lengthier descriptive name, associated Schedule Item, DSS and spacecraft(s). In the future, this task will be handled by the name translation job in Spark instead, so that all monitor data identity transformation and association work contained in a single module, in addition to making use of Spark's fast processing capabilities. With the two Kafka topics, other client software have the option of easily consuming either or both the untransformed real-time monitor data and the transformed. CEP Spark jobs that handle other use cases can simply depend on the topic that carries the name-standardized data.

B. Framing Events in Context

By correlating the planned events data (Schedule Items and SOEs), operator/TDN log data (NMC logs), real-time monitor data, and others (astrophysics data), CEP can provide a more intelligent situational insight than that of simple condition or limit checking. A use case instance that has been demonstrated using the DCEP system is the determination of whether or not a loss of spacecraft signal in the middle of a track is a deviation from the planned service. For this demonstration, the Dawn spacecraft was selected as the candidate due to its frequent occultation around the dwarf planet Ceres (September 2015). During a sample pass, the downlink receiver would record the P_c/N_0 measurement (signal-to-noise ratio) as switching back and forth between -300 (which indicates that the measurement is not valid) to ~25 decibel-Hertz. This measurement is published in real-time as monitor data. The LCO supporting the track would not be startled to see this, since he would have the understanding that this is happening as a result of Dawn orbiting around Ceres. If we took out the human operator from this use case, however, then CEP is needed to indeed verify that the observed data is not a sign of an anomaly. Also, CEP itself can produce another metric that is useful: how accurately the various data correlate to each other.

To apply CEP to this particular scenario, another type of data that the DCEP system needs to process in real-time is the SOE. SOEs for Dawn include the predicted occultation times, both when the spacecraft will enter and exit each occultation. In addition, Dawn's SOEs even include the specific signal loss times as will be observed on Earth. This latter event provides the best accuracy when trying to correlate the SOE data to the P_c/N_0 monitor data's drop to -300 dbHz. To allow DCEP's Spark applications to access this SOE data on the fly, the DCEP team created a Java library that uses SPS's RESTful API to fetch any SOE in the order of seconds. Fetched SOEs are cached in memory until their window for useful real-time correlation expires or new versions are available from SPS. The library also pre-extracts a set of key events listed within the SOEs and makes them available through its API. (These events include symbol rate settings, downlink receiver activation/deactivation, uplink transmitter activation/deactivation, occultation

times, signal loss times, and so forth.) This allows a Spark application that is performing CEP to easily and quickly access the events of interest from any applicable SOE in real-time.

The demonstration showed that the instances where the P_c/N_0 dropped to -300 dbHz in the particular Dawn pass closely followed (in time) the signal loss events predicted in the SOE. Figure 12 shows one sample of such instance from the demonstration. Therefore, the loss of signal observed is not a deviation of service, and the rest of the track can proceed normally. Although this result confirms what the LCOs can already determine fairly quickly during a live pass, it is meaningful because it demonstrates just one building block out of many that can be processed together and correlated to interpret situations within contexts—all without a human operator involved. Also, unless the LCO is very watchful, the fact that there was a time discrepancy of 7 seconds between the SOE's predicted signal loss time and the actual drop to -300 dbHz in the P_c/N_0 measurement may not have been clearly noticed during the track. These sort of metrics determined through CEP—in this case, the differences in predicted versus actual—can provide valuable information. They can indicate whether or not the DSN services are operating optimally and if there exist problem areas that need to be examined.

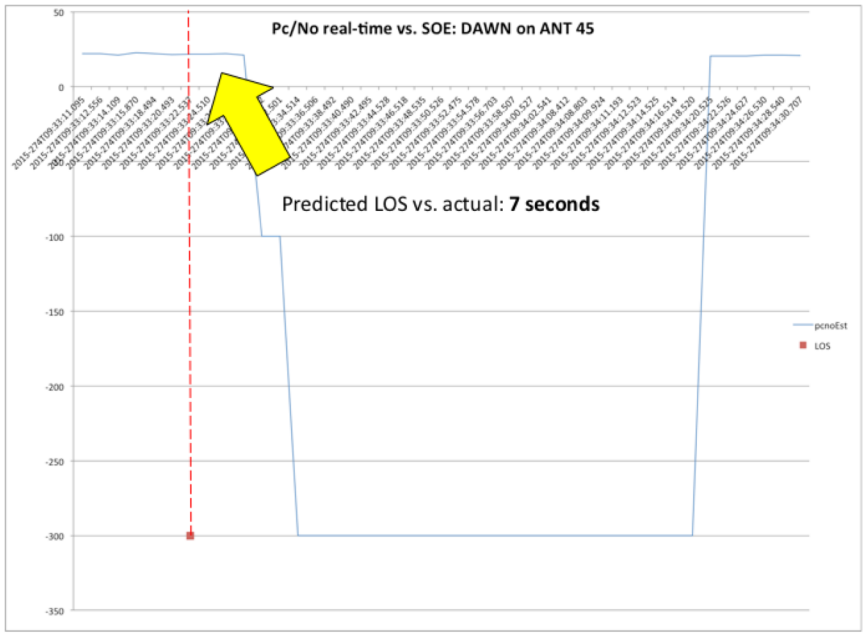

Figure 12. Signal loss predicted in the SOE for Dawn (*LOS*) and the actual loss observed (*pcnoEst* value of -300). Automatic correlation of information from disparately different data sources will frame events in context, resulting in minimal false-positive and false-negative real-time situation assessments.

C. Detecting Deviations from the Norm

Going a step further, by adding historical data to the mix of the complex events processed, the DCEP system can automatically detect deviations from what has been the norm in the past. This opens up many possibilities for doing live statistical analysis and thereby improving operations with the additional intelligence. For example, if a spacecraft track is in progress at the moment, it can be useful to find out if the performance of the current pass is coherent with that of the last pass, given that the two tracks' configurations are similar. This performance can be measured on several terms. If a track that has just ended recorded N number of good telemetry frames during its service duration time of T, then comparing the N/T ratio against the spacecraft's last similar pass (e.g. same DSS), last few passes, or the historical average of such passes in the past Y years will undoubtedly yield useful information about pass performance. For such analysis involving historical data, the DCEP system uses the SQA subsystem's repository via the PL/SQL interface previously mentioned. The interface allows CEP to retrieve those historical metrics that are periodically pre-calculated so that correlations can be performed in real-time or near real-time.

Another type of useful analysis made possible by the DCEP system is the comparison of one element under examination to its other counterparts in the DSN. For instance, comparing the historical performance of one 70-meter antenna to that of DSN's other 70-meter antennas could yield valuable information. Lastly, it is possible to determine trends—generally over time or conditioned by external events—using CEP.

D. Matching Incidents to Known Discrepancies

When the DCEP system is able to use the DRMS repository as a data source and match ongoing situations in real-time to incidents previously captured as DRs or MDRs, this will translate to saving much human effort when anomalies are encountered during operations. In order for this to be possible, the DCEP system will need to be able to understand

the problem symptoms written up in these reports. This presents a huge challenge because the reports are, for the most part, composed in human language (English). The contents of these reports will have to be either translated or encoded in some way in order for the DCEP system to accurately correlate the real-time situation with any recorded incidents in these reports. As one example in this use case, there exists an MDR that documents a recurring problem where the 'Downlink Channel Controller subsystem stops outputting telemetry to the Data Capture and Delivery subsystem.' The report lays out what symptoms the operator should watch for: block count stopping when telemetry is in lock and missing low-criticality progress messages in the NMC logs. Detecting this anomaly alone thus requires three data sources: the MDR repository, monitor data (real-time), and the NMC logs (also real-time). Once the current situation is matched to an existing MDR, the DCEP system can both immediately notify the LCO and—if configured to do so—start the execution of the recovery procedure which is also documented in the report. Much work by the DSN operators have gone into documenting discrepancies that occur during operations. By leveraging upon that work, the DCEP system provides a way for future operational anomalies to be handled in much more effective ways.

E. Generic Tools

In line with the use of open-source software Apache Kafka and Apache Spark, the DCEP system is designed to be an open system. Interoperability through use of open standards and extensibility are two of the design choices made with regard to the DCEP system. This allows the system to be 'tapped into' by other tools, and these beneficiary tools can leverage the messaging and processing already performed by the DCEP system. There currently is a tool in use by the DSN Project that proves this point. As mentioned previously, a new requirement was added to the DSN recently: Track each DSN spacecraft's on-board command-loss timer. Estimating these values requires access to the real-time monitor data. Because the DCEP system was already in place, consuming the real-time monitor data stream, the data was already on the Kafka message bus. From here, two choices could be made: implement the handling of the use case in Spark or create a separate tool that will only use the DCEP system's message bus. Since the CEP engine (Spark) and its applications are still being experimented with (but the command-loss timer tracking is a real project requirement that needs to be satisfied now), the DCEP team created a Java application that consumes the data from Kafka and calculates the timer estimates from that data. (Missions-informed table of timeout values for each spacecraft and astrophysics data—for the OWLT values—make up the other sources of data.) The results are then published every few seconds on JPL's intranet as a web page, so that DSN engineers and managers can readily access the latest information showing the risk of a command-loss timeout event. Because the scalable, distributed messaging infrastructure has been established as a part of the DCEP work, many generic tools such as the Command-Loss Timer Tracker can now be created and immediately access the data streams (both raw and complex-event-processed) with relative ease, extending the possibilities of operational intelligence further.

F. Postage Stamp

The use case to support the Postage Stamp user experience project is demonstrative of the processing that the DCEP system can handle on behalf of the external users, sparing them from the nitty-gritty, low-level nature of the DSN data. Postage Stamp requires real-time data that includes additional context. For example, not only does Postage Stamp require the current azimuth and elevation angle values of a DSN antenna, it also needs to know the antenna's operating state at the moment: on point, slewing, stopped, stowing, stowed, et cetera. The subsystem that controls the antenna movement does not publish these state values. Therefore, determining the real-time operating state of an antenna requires an algorithm. For instance, to determine that the antenna is in the stow position, it is necessary to first obtain its current azimuth and elevation angles, then compare that to the angle values when it is in the stow position (known ahead of time), and also verify that the antenna angles have not changed in the last few seconds (which could indicate that it is slewing or stowing instead). Since there are mechanical parts involved, the algorithm also needs to take into account margins of error in the measurements. As a matter of fact, to complicate things, DSN antennas do not share uniform stow position angles, so the algorithm also needs to handle such differences.

The DCEP team implemented these kinds of algorithmic data processing required by the Postage Stamp inside a Spark application. As the necessary calculations are completed by the Spark jobs (which happens periodically, since it is using the Spark Streaming library), a JSON data object is created and this is made available over the network using a WebSocket. Postage Stamp then consumes this data and reflects the information directly onto its GUI. The DCEP system takes care of much of the cumbersome processing work needed on the raw DSN data. In this way, future technology advancement projects may no longer require expert knowledge of the DSN. The DCEP system can be used to manage the complexity and shield the external users from being affected by DSN's intricacies.

IX. Rule Implementation

In order for the DCEP system to handle the aforementioned use cases and others, the set of logic that performs the actual CEP have to be implemented inside Spark applications. An individual unit of such CEP logic can be referred to as a *rule*. There are two key methods by which real-time decision-making, operator notification (e.g. alarms), and action rules can be implemented in the DCEP system. First, the rule can be based on an expert-driven model of the DSN system, such that threshold-based rules can be executed against incoming data streams and warehoused data repositories to indicate the presence of special conditions. The second method is a data-driven model, where machine-learned relationships between the data constituents of DSN streams and repositories identify the presence of irregularities. The DCEP system will use both methodologies such that existing expert-based knowledge as well as machine-learned relationships can dually play a role in fault detection and correlation analysis.

A. Expert-Based Rules

Based upon decades of successful operation of the DSN, a body of operational threshold-based rules has been built up over the years, upon which real-time monitor data and other types of data can be judged. The DCEP team plans to fully leverage this expert-based knowledge and implement it as a set of rules within the DCEP system's knowledge base, such that the system matches incoming data against relevant rules and makes decisions upon data arrival. Implementation-wise, this knowledge base consists of a set of procedures, which map incoming data points against rule conditionals that generate expert-driven decisions. This implementation model can allow for chaining of rules for more complex conditionals. Although the knowledge base is not necessarily a logical programming inference engine, the ability to add new rules or facts during runtime is not withheld and thus decision-making based on expert-rules can increase in capability over time. In fact, it turns out that implementing logical rules by this method is a sound means to adhere to the DSN operational requirements, where rules are—in many cases—explicitly specified and mandated. An example of such a case is the conditional alarm-sounding requirement for wind speeds at DSN antenna complexes. Each DSN antenna type has specific requirements that dictate safe operational use based on outside wind speeds. These conditions, for example requiring antennas to be driven to stow (pointing to zenith) if outside wind speeds surpass 50 miles per hour, represent well-documented thresholds that require operator action. The DCEP system can automatically match real-time situations to such predefined conditional rules, and it can then provide alerts continuously for all antennas based on such rules being available in the system's knowledge base.

B. Machine-Learned Rules

Where high-quality training data is available and usable,[6] the DCEP system is able to derive machine-learned relationships among constituent data that can help in more accurate automatic decision-making. The key idea is to leverage machine-learning algorithms to train classifiers in mapping expected fault detection scenarios against relevant input data, based upon relationships inherent in the training data. This methodology is in contrast to explicitly programming threshold conditions by which to judge and correlate data. Regression models as well as deep-learning techniques are viable candidates as methods to produce machine-learned, decision-making classifiers. The DCEP team ran a prototype study of the viability of machine-learning for CEP. In this study, over thirty-three million data points were processed to develop a spacecraft identity classifier. The results of this study showed that a machine-learned approach is viable for CEP in the DSN. In many cases, however, the low-quality labeling of training data and the lack of sufficient data points turned out to be a severe hindrance to effective classifiers. Irrespective of the exact machine-learning approach used, one of the DCEP team's goals is to leverage a knowledge base of procedures (in this case, classifiers), which will be invoked during runtime based on real-time data characteristics. In other words, the same high-level interface as the expert-driven rule system will be used so that the machine-learned and the expert-driven conditionals for fault detection can be used interchangeably.

X. Future Work

Much work still remains in order to raise the technology readiness level of the DCEP system so that it can qualify to assume operational responsibilities in the DSN. Upfront, there still remains additional data sources that need to serve as input to the DCEP system: logs (e.g. NMC logs), DRs and MDRs in the DRMS repository, and possibly others that have yet to be identified. Furthermore, to become operations-ready, the DCEP system's existing data input have to be improved to become more robust. This will involve switching from consuming the real-time monitor data via the MON-2/JMS bridge to directly receiving them from the MCIS infrastructure at the DSCCs (in other words, closer to the source). This change will enable key improvements in CEP: the propagation delay from the data source to the DCEP system will be minimized, the entire set of monitor data at the DSCCs will be available for subscription (whereas via the MON-2/JMS bridge, only a limited set can be subscribed), and the DCEP system will receive

streaming data at its highest rate (whereas via the MON-2/JMS bridge, each individual monitor data item is metered to five seconds). Making this switch, however, will require careful engineering to make sure that the existing DSN services are not adversely affected by the introduction of the DCEP system in the operational ecosystem. Also, as the use cases become more comprehensively defined, the interface between the DCEP system and the SQA subsystem will undergo refinements, so that additional historical metrics can be made available for CEP. All of this work is critical and, at the same, extensive.

It will be important for the operational DCEP system to work around the unavoidable WAN bandwidth limitation that exists in the DSN. As mentioned, the strategy for overcoming this challenge is to process all voluminous DSCC data locally, at the DSCC itself, and transmit only intelligent information that are smaller in size to the other DSCCs and NOCC. The ultimate goal is to enable FtSO this way. Design-wise, this strategy entails two levels of CEP: CEP being done at the local DSCCs and a global CEP that processes key events from the entire DSN. Forming the requirements and design for this multi-level CEP in the DSN is another area of work that will be done in the future.

SQA is an excellent subsystem that warehouses a treasure trove of historical DSN monitor data that is valuable for CEP, but there never was a requirement placed on the SQA for it to serve as a data provider to the DCEP system. The current PL/SQL interface was designed to minimize the burden on the SQA subsystem when the DCEP system accesses its data. It is now clear that this is a big data problem, and for the DCEP system to make more flexible use of the historical data, a different solution may be needed. For instance, rather than using the SQA subsystem itself as a CEP data source, a separate big data database (e.g. Apache Cassandra) can be created and initialized with a copy of SQA repository's data. As the DCEP system consumes real-time monitor data from the DSN, the data is also loaded into this new database, thereby keeping the historical data constantly up to date. With this new elastic database, the DCEP system will be able to access monitor data from the past and its statistics with greater freedom and with better performance.

Another area of future work is devising a novel way for managing the CEP logic, or rules. As mentioned, there exists a number of rule-based systems in the enterprise industry for doing business intelligence. The DCEP team will investigate and glean from the solutions for rule management that already exist in these systems. Most likely, due to the unique set of requirements and operational nature of the DSN, a customized strategy will have to be devised for managing the CEP rules in the DSN.

Finally, it is important to ascertain the performance capabilities of the DCEP system as a whole as well as that of its individual parts (i.e. Spark, Kafka, et cetera). Having the understanding of the system performance and its limitations will inform further design choices for the DCEP system. To that end, the DCEP team plans to establish testing scenarios that push the CEP and the messaging infrastructure to their limits, to get a clearer picture of what is possible (and what is not) with the DCEP system.

XI. Conclusion

The DSN currently provides reliable, high-quality services to the missions that it supports, but further improvements need to be made to its operational cost-effectiveness. Using complex event processing, greater intelligence can be extracted from operations in real-time: anomalies can be detected quickly; false positives and false negatives can be significantly reduced through the association of contexts; using the knowledge base of previously documented discrepancies, newly encountered ones can be automatically labeled and remediated directly by the DCEP system; and through machine learning, the DCEP system will train itself to detect events even smarter as time goes by. These benefits are in addition to the immediate, more basic solutions that the DCEP system provides, such as normalizing the irregularities in DSN's data identifiers, deriving the higher-level state information through algorithms for outside systems like Postage Stamp, and so forth. Further development, prototyping with more use cases, and testing need to be done in order for DCEP to be formally accepted as part of DSN Operations; but so far, the progress up to this point indicates that CEP is a truly viable solution that will help usher DSN Operations into the future and serve as a key enabler for the Follow-the-Sun Operations.

Acknowledgments

The authors thank: Bach X. Bui, Silvino C. Zendejas, Ara Kassabian, Michael A. Rueckert, and Saman Saeedi for their valuable contributions to the DCEP task; and Dr. Alexandra Holloway for providing the sample graphic of Postage Stamp. The authors also give special thanks to Michael E. Levesque and Jay E. Wyatt for their support and sponsorship of the task. The aforementioned individuals are all affiliated with JPL.

The work described in this paper was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech), under a contract with the National Aeronautics and Space Administration (NASA). The work was funded by the Space Networking and Mission Automation Tech Program.

References

[1]Office of Audits, "NASA's Management of the Deep Space Network," Office of Inspector General, Report No. IG-15-013, 26 Mar. 2015.

[2]Johnston, M. D., Levesque, M., Malhotra, S., Tran, D., Verma, R., and Zendejas, S., "NASA Deep Space Network: Automation Improvements in the Follow-the-Sun Era," *24th International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina, 2015.

[3]Davenport, T. H., "The Confusing Landscape of 'Cognitive Computing'," *The CIO Report (The Wall Street Journal)*, 17 Dec. 2014, URL: http://blogs.wsj.com/cio/2014/12/17/the-confusing-landscape-of-cognitive-computing/?mg=id-wsj.

[4]Springer, K., "British Tourists' Tweets Get Them Denied Entry to the U.S.," *TIME Newsfeed*, 31 Jan. 2012, URL: http://newsfeed.time.com/2012/01/31/british-tourists-tweets-get-them-denied-entry-to-the-u-s/.

[5]Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., Stoica, I., "Spark: Cluster Computing with Working Sets," *2nd USENIX Workshop on Hot Topics in Cloud Computing*, Boston, MA, 2010.

[6]Jones, N., "Computer science: The learning machines," *Nature*, Vol. 505, No. 7482, Jan. 2014, pp. 146–148.