Statistical risk analysis and the development of robust distributed applications for NASA's Deep Space Network

Ian W. Bornhoeft,[1,2,*] Silvino C. Zendejas,[1,†] and Paul A. Wolgast[1,‡]

[1] *Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr, Pasadena, CA 91109*
[2] *Department of Physics, Applied Physics, and Astronomy, Rensselaer Polytechnic Institute, 110 8th St, Troy, NY 12180*

(Dated: August 19, 2013)

Deep Space Network (DSN) equipment failures can cripple NASAs telecommunication service capabilities and result in irreversible data losses during mission nominal and critical events. The continuous effort to make the DSN more operationally efficient and less costly to operate, introduces challenges in the ground data system software architecture and the ability to manage risk effectively. The concepts behind relational databases and the tools provided by the Structured Query Language (SQL) in Oracle, provide a means for quantifying the risk of data loss under various conditions involving different levels of automation. The concepts behind the development of the Erlang programming language, designed to address the needs of robust, reliable, and efficient telecommunication software solutions, can be applied to specific functional areas of the DSN Service Management System (DSMS). Using a combination of predictive methods based on analyses of historical DSN error data and more robust distributed software architectures developed with Erlang, it is possible to make the DSN more reliable and operationally efficient.

I. INTRODUCTION

The Deep Space Network (DSN) has been in operation since 1959, and most of its antennas were built in the 1960s and 1970s. Although five new antennas were added in the 1990s and one in 2003, the DSN is aging and, combined with the fact that the DSN is supporting more missions at further distances from the Earth than it was originally designed to handle, equipment failures, while still remaining relatively infrequent, have become more incessant and detrimental in recent years. [1] [2] It is therefore more necessary than ever for methodologies to be in place which can handle such failures when they occur.

In this investigation, we develop methods to analyze historical DSN data to determine the probability of failure for a given process, the likelihood that telemetry data is lost, and a reasonable estimation for the severity of such failures if they do indeed occur. We then go on to present new methods, using the Open Telecom Platform (OTP) Design Principles developed as part of the Erlang programming language, for implementing more robust messaging software solutions within the DSN.

*Electronic address: bornhi@rpi.edu; URL: http://www.jpl.nasa.gov
†Electronic address: silvino.c.zendejas@jpl.nasa.gov; URL: http://www.jpl.nasa.gov
‡Electronic address: paul.a.wolgast@jpl.nasa.gov; URL: http://www.jpl.nasa.gov

II. RISK ANALYSIS

By querying DSN data placed on an Oracle database server called SERVICELOSSDATASET with Structured Query Language (SQL) and performing statistical analyses on the queried data we were able to develop general and robust methods for determining the likelihood, severity, and duration of various errors and failures with respect to a number of different variables which were each entered into the database as a column. These variables included:

SCHEDULE_ITEM_ID a unique identification number for each DSN event

MISSION identifies which mission is most directly involved with the specified event

ANTENNA identifies which DSN antenna is involved with the event

DSCC (Deep Space Communications Complex) identifies the DSN complex involved witht the event

ASSEMBLY identifies the DSN assembly involved with the event

TLM_SERVICE_LOSS_MINUTES identifies how much telemetry was lost (in minutes)

In order to gain an understanding of the difficulties facing the DSN and their implications it was necessary to calculate several statistically significant quantities for the set of historical DSN data with which this project was presented. These quantities included the number of events for a particular value of MISSION, ANTENNA, DSCC, or ASSEMBLY, the number of those events which resulted in a failure, the probability of failure, the probability of telemetry loss due to such a failure, the average

time lost per failure resulting in data loss, and the time loss values of several different percentiles formed by the distribution of time losses.

Due to the nature of the primary author's background as a physicist the first approach taken to attempt to tackle this problem was to search for elegant and exact mathematical solutions to the questions posed. The thing that seemed most obvious to try first was to attempt to determine if this data fit a common (or uncommon for that matter) normalizable probability distribution, and if it did to identify the form of this distribution and determine any unknown parameters associated with the particular distribution.

First, the expectation value of the data was calculated as follows:

$$\bar{x} = \sum_{i}^{N} x_i p_i \approx 5.037 \text{ minutes}$$

Where x_i is the ith telemetry loss and p_i is the probability of that loss.

The calculations for the probability and mean value have not been included as they are trivial. However, the mean value (μ) of the telemetry loss is approximately 5.037 minutes as well, which was encouraging, as it meant that this was more likely to be an intuitive and somewhat symmetric distribution.

From an inspection of the frequency distribution (Figure 1) of the data with respect to telemetry lost it was immediately evident that if the DSN data were to fit a normal distribution it must have an enormous negative skew.

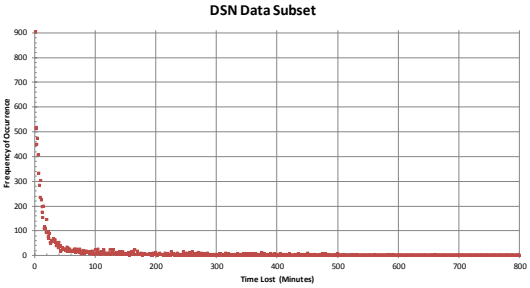

FIG. 1: This is a plot of the presumably representative subset of DSN event data provided to me, excluding events where no telemetry service was lost (or at least much less than one minute was lost) so that the shape of the curve that is seemingly traced out by the data plot can be seen with the naked eye.

The following was our attempt to calculate the skewness of the DSN data:

$$\gamma_1 = \frac{E[(X-\mu)^3]}{(E[(X-\mu)^2])^{3/2}} \approx 10.497$$

After attempting to calculate the skewness of the data, and obtaining a positive value, we determined that this data could not possibly tend toward a normal distribution. We then attempted to fit the DSN data to a variety of other distributions, including some invented solely for this purpose. Unfortunately, none of these normalizable distributions quite fit with the data. Eventually a function was found that did seem to fit the data if the cases in which there was no telemetry loss were ignored. It seemed reasonable that perhaps this point was an outlier and could simply be omitted. Unfortunately, although it was possible to fit the data to a curve, this curve happened to be a variant of the hyperbolic cosine or hyperbolic cosecant functions, neither of which is normalizable over any reasonable intervals.

Even though we were not able to fit the DSN data to a continuous distribution, we were able to make headway using slightly less elegant numerical methods, which, in general, are better suited to deal with discrete probability distributions. We devised these methods in ways which were both easily implementable in SQL and very general. These methods can be easily modified to answer a wide range of additional questions which may be asked in the future. We were able to calculate all of the required information with relative ease.

Originally, we wrote functions to calculate the number of events, number of failures, probability of failure, probability of data loss, average telemetry loss, and ninety-fifth percentile telemetry loss for each given assembly. After realizing that it would be advantageous to devise these methods in a more genral form, we modified these functions to receive assemblies, antennas, DSN complexes, or missions, and calculate the same statistics. For the purposes of making graphical representations of the query results, we realized that it would be useful to calculate the fifth percentile of telemetry loss values as well as the ninety-fifth, so that we could use error bars to show the range that ninety percent of the data fell into, which helped to give a sense of how uniform the losses due to failures were for a particular component. As a result of this realization, we generalized my ninety-fifth percentile function to return any given percentiles value (Appendix A). The concepts behind relational databases and the tools afforded SQL language made the implementation of such methods relatively simple. As a result, the original goals of our project were accomplished in a considerably shorter amount of time than was expected.

III. ROBUST AND COST EFFECTIVE SOFTWARE SOLUTIONS

Upon the completion of the original project goals we explored methods for making the DSN software more robust to failures given that software is a large contributor to service interuptions. We turned to the language of Erlang since we felt that it might have significant advantages, at least in this context due to its OTP Design Principles, over other solutions given that Erlang was designed to respond to the requirements of telecommunication system applications. [3]

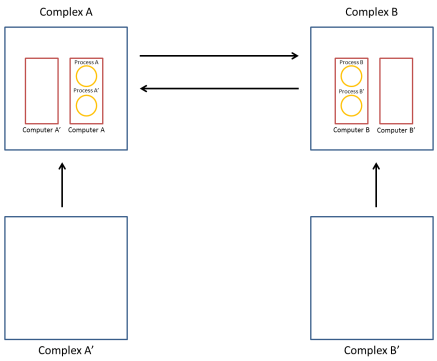

FIG. 2: This is a diagram of our particular conceptual implementation of the OTP Design Principles to work with the DSN.

The main concept of our procedures is illustrated in Figure 2. Two different locations which are part of the DSN (labeled as $ComplexA$ and $ComplexB$ in Figure 2) can directly communicate with one another. In the case of an error at $ComplexA$, our concept entails that $ComplexB$ should not be affected, or even otherwise be aware of the failure. The idea is that there should be another process already running ($ProcessA'$) which is capable of seamlessly taking over without losing any information in transit between $ComplexA$ and $ComplexB$. Using the OTP Design Principles The information in transit can be put into a queuing process which can store the information until $ProcessA'$ has successfully taken over the responsibilities of $ProcessA$. [5] Furthermore, if for some reason $ComputerA$ has a hardware malfunction, or some other kind of irrecoverable failure, $ComputerA'$ should be ready and able to seamlessly take over. This can even be implemented on a larger scale by completely separate complexes. If $ComplexA$ suffers a fatal error, $ComplexA'$ can take over without disturbing $ComplexB$ at all. This methodology need not be limited to one back-up system at each level. It may be prudent in many cases to have $ProcessA''$ or even $ProcessA'''$ in situations in which errors are particularly likely.

All of these things can be accomplished within the framework provided by the OTP framework of Erlang using its distributed programming implementation with supervisor processes. [4] When a child process encounters an irrecoverable error it notifies its supervisor and ends itself. Its supervisor can then restart the child process. When a supervisory process encounters an error it stops all of its child processes and then itself. This behaviour can be modified to cause the supervisor to start a new version of itself before stopping, which would then go on to respawn its child processes. This kind of a framework allows for very robust networking which is capable of isolating and eliminating problems quickly, efficiently, and, most importantly, without affecting the rest of the network.

IV. CONCLUSIONS

During the course of this project failures of the DSN were investigated. Using a subset of historical DSN event data as a statistical case study, general SQL queries were developed and tested for extracting meaningful information from the overwhelming amount of raw data generated by the DSN. The original intent of this investigation was simply to provide the DSN administration with quantitative measurements of the DSN's shortcomings, as well as to arm them with the tools to easily reassess the DSN in the future. This goal was accomplished.

The project then evolved into an exploration of the applicability of the Erlang language OTP framework to the design of more robust DSN software solutions. Time did not allowed for a direct comparison between Erlang and alternative architectures written in Java or C/C++ which are currently the languages primarily used in the DSN. However, the authors of this paper are fairly confident that Erlang provides a viable option for the design and implementation of selective applications within the DSN software infrastructure.

[1] Bagri, D.S., Statman, J.I., Gatti, M., in *Proceedings of the IEEE*, vol. 95, no. 10, Pasadena, 2007.
[2] Wackley, J., Dundics, D., The Interplanetary Network Progress Report 41-187, 2011.
[3] Armstrong, J., Ph.D. thesis, Royal Institute of Technology, Stockholm, Sweden, 2003.
[4] Logan, M., Merritt, E., Carlsson, R., *Erlang and OTP In Action* (Manning Publications Co., Stamford, CT, 2011).
[5] Cesarini, F., Thompson, S., *Erlang Programming* (O'Reilly Media, Inc., Sebastopol, CA, 2009).

Acknowledgments

This work was supported by the California Institute of Technology and the National Aeronautics and Space Administration. I.B. thanks Harihar Subramanyam, Cecilia Sanders, and Michael Younkin for their technical assistance, as well as David Mohr for his guidance and unreasonable kindness.

APPENDIX A: QUERY RESULTS FOR A SAMPLE DATA SET

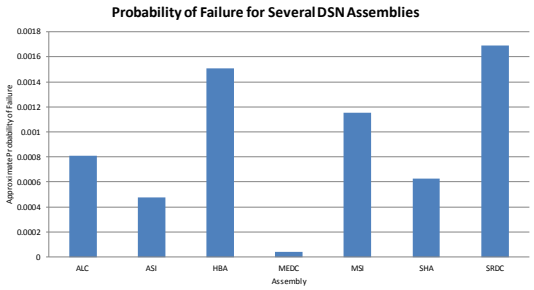

FIG. 3: This graph displays the probability of failure for several DSN assemblies.

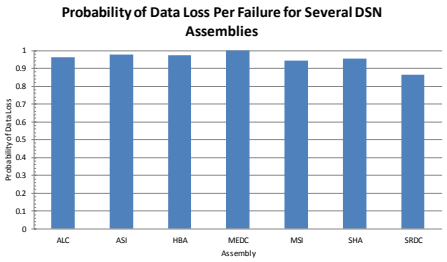

FIG. 4: This graph displays the probability of data loss per failure for several DSN assemblies.

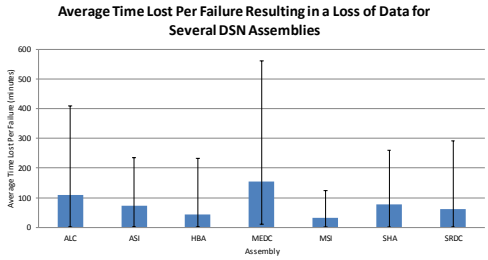

FIG. 5: This graph displays the average data loss per failure resulting in data loss for several DSN assemblies in minutes. The error bars denote the 5^{th} and 95^{th} percentile values.

FIG. 6: This graph displays the probability of failure by DSN mission.

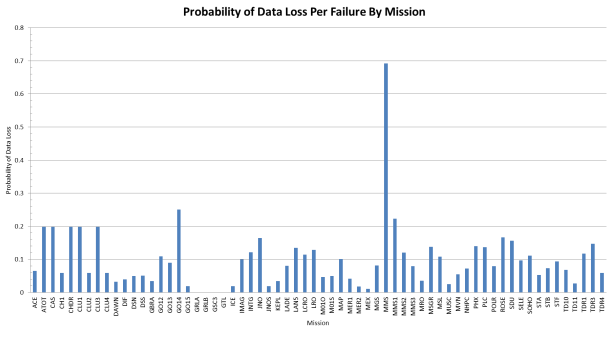

FIG. 7: This graph displays the probability of data loss per failure by DSN mission.

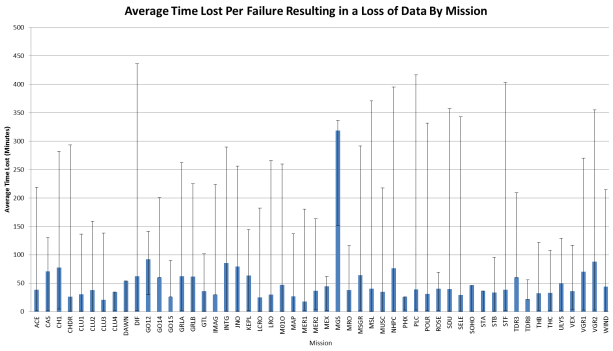

FIG. 8: This graph displays the average data loss per failure resulting in data loss by DSN mission in minutes. The error bars denote the 5^{th} and 95^{th} percentile values.

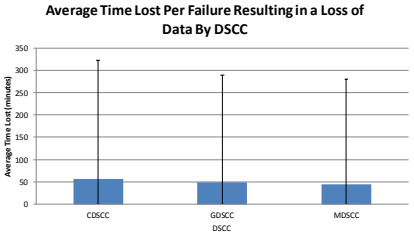

FIG. 11: This graph displays the average data loss per failure resulting in data loss by DSCC in minutes. The error bars denote the 5^{th} and 95^{th} percentile values.

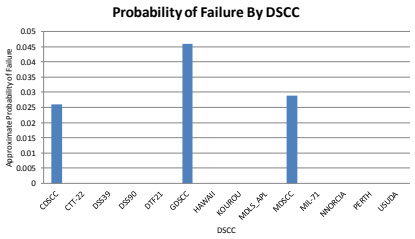

FIG. 9: This graph displays the probability of failure by DSCC.

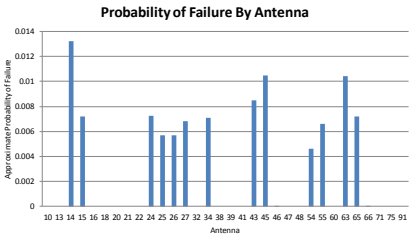

FIG. 12: This graph displays the probability of failure by DSN antenna.

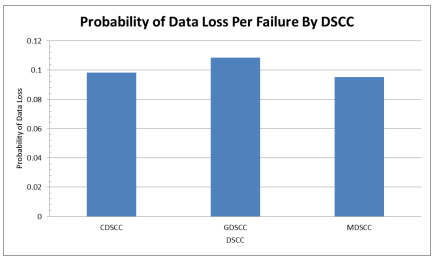

FIG. 10: This graph displays the probability of data loss per failure by DSCC.

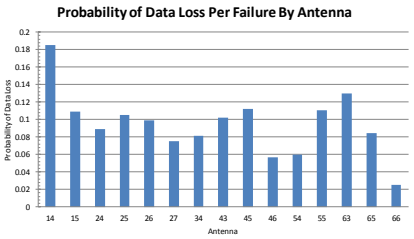

FIG. 13: This graph displays the probability of data loss per failure by DSN antenna.

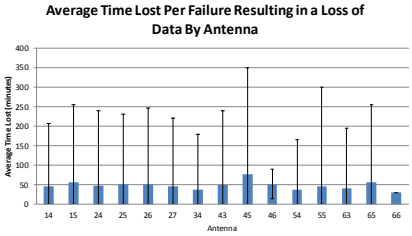

FIG. 14: This graph displays the average data loss per failure resulting in data loss by DSN antenna in minutes. The error bars denote the 5^{th} and 95^{th} percentile values.