

Navigation Ground Data System Engineering for the Cassini/Huygens Mission

R. M. Beswick^{*}, P. G. Antreasian[†], S. D. Gillam[‡], Y. Hahn[§], D. C. Roth^{**}, and J. B. Jones^{††}
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109 USA

The launch of the Cassini/Huygens mission on October 15, 1997, began a seven year journey across the solar system that culminated in the entry of the spacecraft into Saturnian orbit on June 30, 2004. Cassini/Huygens Spacecraft Navigation is the result of a complex interplay between several teams within the Cassini Project, performed on the Ground Data System. The work of Spacecraft Navigation involves rigorous requirements for accuracy and completeness carried out often under uncompromising critical time pressures. To support the Navigation function, a fault-tolerant, high-reliability/high-availability computational environment was necessary to support data processing. Configuration Management (CM) was integrated with fault tolerant design and security engineering, according to the cornerstone principles of Confidentiality, Integrity, and Availability. Integrated with this approach are security benchmarks and validation to meet strict confidence levels. In addition, similar approaches to CM were applied in consideration of the staffing and training of the system administration team supporting this effort. As a result, the current configuration of this computational environment incorporates a secure, modular system, that provides for almost no downtime during tour operations.

Nomenclature

<i>CIA</i>	= Confidentiality, Integrity, and Availability
<i>CIS</i>	= Center for Internet Security
<i>CISscan</i>	= CIS internal host security scanner
<i>CM</i>	= Configuration Management
<i>DOS</i>	= Denial of Service attack
<i>DSN</i>	= Deep Space Network
<i>DR</i>	= Disaster Recovery
<i>ECC</i>	= Emergency Control Center
<i>GDS</i>	= Ground Data System
<i>GCC</i>	= Goldstone Communications Complex
<i>HP-UX</i>	= Hewlett-Packard (Unix System V based OS) for Hewlett-Packard computers

^{*} Cassini Navigation Senior System and Network Administrator/ISCS System Engineer, Section 343, Guidance, Navigation, and Control, 4800 Oak Grove Dr., M/S: 230-205, Pasadena, CA 91109, USA, AIAA Member.

[†] Cassini Orbit Determination team lead, Section 343, Guidance, Navigation, and Control, 4800 Oak Grove Dr., M/S: 230-205, Pasadena, CA 91109, USA, AIAA Member.

[‡] Cassini Optical Navigation team lead, Section 343, Guidance, Navigation, and Control, 4800 Oak Grove Dr., M/S: 230-205, Pasadena, CA 91109, USA, AIAA Member.

[§] Cassini Maneuver Analysis team lead, Section 343, Guidance, Navigation, and Control, 4800 Oak Grove Dr., M/S: 230-205, Pasadena, CA 91109, USA, AIAA Member.

^{**} Cassini Navigation Deputy team chief, Section 343, Guidance, Navigation, and Control, 4800 Oak Grove Dr., M/S: 230-205, Pasadena, CA 91109, USA, AIAA Member.

^{††} Cassini Navigation team chief, Section 343, Guidance, Navigation, and Control, 4800 Oak Grove Dr., M/S: 230-205, Pasadena, CA 91109, USA, AIAA Member.

<i>IDE</i>	= Integrated Drive Electronics disk drive type
<i>IGNITE</i>	= System imaging and installation software for HP-UX systems
<i>ISS</i>	= Internet Security Systems network security scanner from Internet Security Systems Inc.
<i>LAN</i>	= Local Area Network
<i>Linux</i>	= Open Source OS derived from Unix
<i>MTTR</i>	= Mean Time to Restore
<i>MMNAV</i>	= Multi-Mission Navigation operations coordinating organization
<i>MMNAV Nav Ops Net</i>	= MMNAV Navigation Operations Network
<i>NAS</i>	= Network Attached Storage
<i>NESSUS</i>	= Nessus network security scanner from Tenable Network Security Inc.
<i>NFS</i>	= Network File System
<i>ORT</i>	= Operational Readiness Test (also known as Operational Readiness Training)
<i>QoS</i>	= Quality of Service
<i>RAID</i>	= Redundant Array of Inexpensive Disks
<i>RAID-4</i>	= RAID array design using a parity disk in the array
<i>RAID-5</i>	= RAID array design using parity information spread across array
<i>RLOGIN</i>	= Remote Login
<i>RCP</i>	= Remote Copy
<i>RSYNC</i>	= Remote Synchronization (file distribution) program
<i>RPM</i>	= Red Hat Package Manager
<i>SATA</i>	= Serial Attached ATA –Advanced Technology Attachment disk drive type
<i>SCSI</i>	= Small Computer System Interface disk drive type
<i>SAS</i>	= Serial Attached SCSI disk drive type
<i>Solaris</i>	= Sun (Unix System V based OS) for Sun computers
<i>SSH</i>	= Secure Shell secure communications replacement for RLOGIN, other “R” commands
<i>SYSTEMIMAGER</i>	= System imaging and installation software for Linux computers
<i>TELNET</i>	= Remote Terminal program
<i>TMR</i>	= Triple Modular Redundant (three redundant components for every point of failure)
<i>TRIPWIRE</i>	= Cryptographic file system validator

I. Introduction

Spacecraft Navigation for the Cassini/Huygens mission involves the processes of Trajectory Analysis, Optical Navigation, Orbit Determination, and Maneuver Design, carried out by teams of engineers under rigorous requirements for accuracy and completeness, often under critical time pressures. Numerous activities required these teams to perform detailed analysis of complex spacecraft data sets, review results and process resultant navigation computations in a rapid and efficient manner, returning results in under one day, in some cases within a two to three hour period with little margin for error. These results then had to be converted, distributed, correlated against further data from the spacecraft, uplinked to the Cassini spacecraft and archived in a complex interplay between the Cassini Spacecraft Navigation team, Science team, Spacecraft Operations team, and other parts of the Cassini Project. (The Orbit Determination estimation processes are discussed by Antreasian, *et al.*¹, and the Maneuver Design orbit control operations are discussed by Williams, *et al.*².) This effort was performed on the workstations, servers and networks utilized for spacecraft operations on the ground, termed the Ground Data System. Requirements to support this navigation function presented a clear case for a fault-tolerant high-reliability/high-availability computational environment to support these data processing needs. Starting out in an *ad hoc* manner, from these performance and reliability constraints, a program of workstation, network and file system benchmarking had evolved over time to support these needs. This program finally culminated in a formal system engineering process for Saturn Tour operations that served the design, implementation, and deployment of a high-performance and reliable computational environment for the Navigation team.

This program involves several associated elements in its design process. Configuration Management became over time an integral and critical element in the system design. Software and hardware components were rigorously standardized, specifying a clearly defined computational environment that had precise controls for which operating system and software sets would be installed on which specific hardware platforms. Furthermore, a greatly clarified system model simplified administration by ensuring that each machine had a well-defined configuration with clearly

understood interoperations with other computational components. The importance of this approach cannot be overstated. In time it allowed complex tasks that used to take hours or days (on the same hardware) to be finished in minutes as well as greatly improving troubleshooting capabilities during system faults and vastly improved abilities for the update, repair and deployment of new computational nodes.

Fault tolerance is integral to the system configuration. Making certain that the system is reliable enough to be used and available to the Navigation team was critical to this effort. Requirements for Quality of Service (QoS) and Mean Time to Restore (MTTR) were formalized and incorporated into the formal design process, and fault tolerance and speed of restoration in the event of a fault became central system engineering constraints.

As a part of these reliability concerns, security was considered to be another aspect of reliability in supporting the Navigation effort in this environment. A system that is hardened to be fault-tolerant against intelligent actors will often prove robust against numerous “natural” random failures as well. Moreover, no matter how accurate the results generated by the computational system, if those results were modified by a security compromise, then whether the system produced the right results would be of no use to the Navigation team. This particular security model, based on the security principles of Confidentiality, Integrity, and Availability (CIA), was used as a framework for system hardening. As a part of this process, security benchmarks and standardized testing tools along with other regression testing validated that the computational environment met certain confidence levels.

Similar approaches to those used in Configuration Management were utilized in consideration of the staffing and training of the system administration team supporting this effort. Clearly defining job roles and tasks, as well as organizing staffing for critical events, such as maneuvers and encounters, served to greatly improve efficiency. An additional benefit was helping to keep a limited system administration staff on a regular sleep schedule! This allowed resources to be focused to resolve the most crucial problems in the absolute minimum time while enabling long term planning to consider ongoing means to further automate navigation operations. The end result was a modular environment that promoted a well-formed system configuration that was secure, easy to administer, and easily allowed global changes across the whole environment – while necessitating almost no downtime during tour operations.

This paper documents these efforts, covering the evolution of the design process from its pre-launch *ad hoc* configuration to the formal design efforts that involved the preparation for Saturn Orbital Tour Operations. This discussion also portrays this effort compared with industry “best practices” for fault tolerance, disaster recovery, and security. We provide an example for similar engineering efforts. While this may be of use to another highly focused Ground Data System for a flagship class interplanetary mission, there is significant applicability for other cases where a large engineering team has to analyze and process large amounts of data in a precise, efficient, and secure manner under tight time constraints.

II. History

Lately it occurs to me: What a long, strange trip it's been, ...
-The Grateful Dead³

In order to understand the evolution of this system engineering process, it may be helpful to consider this effort against the overall background of the Cassini/Huygens mission. The Cassini/Huygens mission launched on October 15, 1997, and spent the next seven years on a journey crossing the solar system until its arrival and entry into Saturnian orbit on June 30, 2004. This joint project between the United States National Aeronautics and Space Administration, European Space Agency, and Italian Space Agency (NASA/ESA/ASI) would perform a total of four planetary flybys on its journey to Saturn, including two Venus flybys on April 26, 1998 and June 24, 1999, an Earth flyby on August 18, 1999 and an impressive dual science mission, culminating during its flyby of the planet Jupiter with the Galileo spacecraft (then in orbit around Jupiter) on December 30, 2000. When the spacecraft arrived in Saturn orbit, the Cassini/Huygens combined spacecraft began a four-year aggressive tour of the Saturnian system, including the deployment of the Huygens probe on December 25, 2004 and the subsequent landing of the Huygens probe on the surface of the Saturnian moon Titan on January 14, 2005. This orbital tour would also comprise more

than seventy orbits of the Saturn system, involving fifty-two close targeted flybys of the largest moons of Saturn, consisting of Titan (comprising the majority of the flybys), Enceladus, Hyperion, Dione, Rhea, and Iapetus. Figure 1 describes the mission from Launch till Saturn Orbital Insertion (SOI).⁴

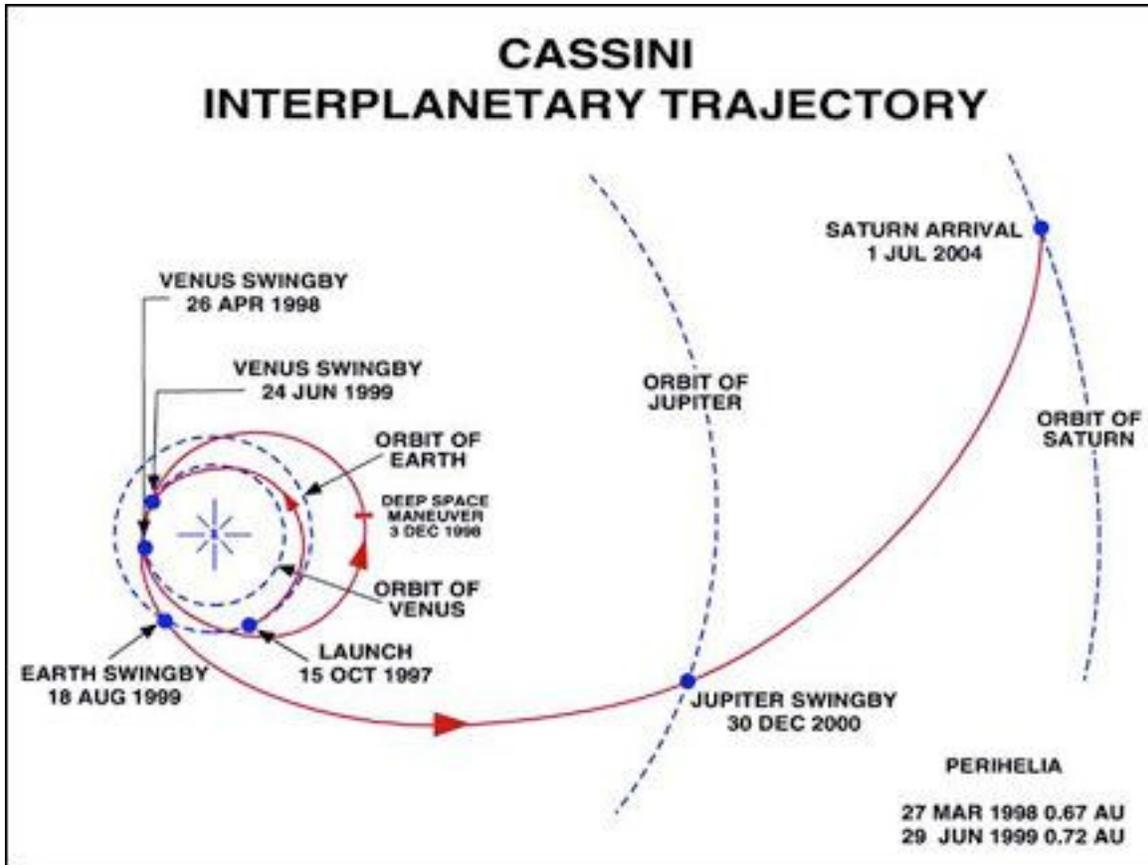


Figure 1. Cassini Launch and Cruise

A. Pre-Launch Environment

Initially, the computational environment for the Navigation Ground Data System, called the Multi-Mission Navigation Operations Network (MMNAV Nav Ops Net for short) suffered from a number of issues that concerned Cassini Navigation Operations. While maintained by very skilled systems and network administrators, this environment had been subject to several years of harsh cost constraints in terms of both computer and network hardware and adequate personnel support for Ground Data Systems engineering. One year prior to launch, an effort was undertaken to revamp, upgrade, and improve the operational capability of this computational environment. At first, this was done in an unstructured manner where immediate concerns dominated such efforts. However, after several years of successful improvements and lessons learned from such efforts, a formal design process was utilized to perform a complete overhaul of the Navigation Ground Data system.

The initial computational system, utilized during the mission design stages prior to launch, consisted of a dozen Hewlett-Packard workstation-class machines and two Sun Microsystems workstation-class machines running on a heterogeneous Ethernet Local Area Network (LAN) scattered across three buildings. Although supported by a well trained network and system administration staff and well documented, this network, among the oldest at the Jet Propulsion Laboratory, had developed in an unplanned fashion. In addition, security and version control of software on the Cassini Navigation computational environment was implemented in an *ad hoc* manner that was confusing and

inconsistent. The result was a configuration that made it difficult to lock down machines, to operate in a secure manner, or determine the cause of a crash. It was clear that improvements needed to be made.

During this pre-launch period, initial fixes and improvements to the Navigation computer environment began to resemble a rather large game of “whack-a-mole,”⁵ for as one problem would be resolved, two (or more) would be revealed. However, a long-term push on the part of the system administration staff during this struggle for better configuration management, fault tolerance, and security would slowly promote a more reliable system design.

Starting in late 1996, one of the first areas that would benefit from this long term effort would be version control—both for operating system releases (and attendant patches) and software versions. Initially the workstations used a variety of operating system releases (HP-UX 9.05 and 9.07 on the HP workstations, Solaris 1.1.3 and 1.1.4 on the Sun workstations) with a varying set of patches applied to each. Software versions varied on each workstation as well, so that it was difficult to determine easily which software set was running under which OS and patch combination. To make matters worse, several Navigation staff members expressed a strong desire to expand cross-mounting between all the workstations to help provide a workaround for this version control issue—so that all conflicting versions of the software could be run on all machines. Instead, what was needed was a single “known good” working version of the software, not twenty different problematic software versions that worked in slightly divergent ways. Additionally, with the cross-mounting envisioned (especially considering the implementation of the Network File System (NFS) file system under HP-UX), if one workstation failed it could bring all the other workstations down. This helpful, but misguided approach, demonstrated one of the first needs for an overarching organizational paradigm change in how this network would be set up.

The resolution of this issue would take many months, but ultimately it served as a catalyst for further design improvements. Each of the workstations on the team would have to be reinstalled with the identical set of operating system software (HP and Sun) and patch clusters and regression tested to ensure that the Flight Software running on the workstations would function as intended to design specifications. The two most powerful workstations (one of which was set up already as a file and compute server) would be set up as identical file servers in a classic “star” configuration, with one running as a warm backup and would serve as the single repository for all software, including Flight Operations ground software, not directly a part of the operating system of the other machines. The cross-mounting approach was opposed vigorously by the Navigation system administration team until other Navigation staff members came to appreciate the advantages of a centralized software repository. The configuration of the machines (with most of the software being offloaded transparently onto a remote file server) was similar to an approach first considered at Carnegie-Mellon University in its Andrew Distributed Computing Environment.⁶ With such a setup, all machines could have their (non operating system) software updated in a single, atomic (as used in Distributed Systems literature)⁷ operation; only one single point of failure (the active file server) would be necessary. This was far superior to each workstation, domino-like, acting as a single point of failure. Additionally, from this effort came the idea that each workstation should be capable of running all software necessary for Navigation operations (i.e. be interchangeable with all other Navigation workstations). This idea was divergent from nominal Flight Operations ground software practice because rather than having a customized set of software for an individual machine, each machine could run *all* of the software, so that in event of a failure, or even just a desire to run software on a different, perhaps faster, machine, each machine *would behave in the same manner*. (As during the Industrial Revolution, the concept of interchangeable parts improved efficiency and would prove invaluable for troubleshooting problems as well as greatly simplifying configuration management throughout the rest of the mission.)

Coming fresh on the heels of this difficult change, it had become clear that the security setup of these machines would need to undergo a similar organizational push. Although SSH and Encrypted TELNET were available, most users still defaulted to using the non-encrypted versions of RLOGIN and TELNET due to familiarity and the fact that the settings for SSH were not the same on each machine. Moreover, the security setup of the machines needed a similar Configuration Managed approach to ensure that all of the machines were tightened down to the same security level. This would involve another standardization effort applied to the machines to ensure that these encrypted communications tools worked as expected, as well as a strong effort towards Navigation staff education on how to use SSH (as a replacement for the set of “R” commands such as RLOGIN and RCP). After some further effort it was possible to totally disable these non-encrypted tools on the Navigation computer systems.

B. Launch Mode

Once this initial round of environmental fixes was accomplished, a significant mission milestone was achieved. The timeline of the Cassini/Huygens Spacecraft would assert itself in 1997, in that the final phases of Launch Reviews and Operational Readiness Test exercises (ORT's) would come to consume all of the available time of the Navigation team, and require a great deal of system administrator effort to ensure that the then-current system configuration would work as expected to ensure a reliable environment for launch. At this point, during these ORT exercises, the embedded system administrator staff would have not just the concern of support of a complex network of production UNIX workstations and servers, but also to experience the demands and concerns of being part of a flight operations team, where the highest possible level of effort would be demanded, at the highest level of priority, where troubleshooting and successful resolution of problems as they occurred, by any means necessary, could be critical to the operation of the Spacecraft. A former Flight Director for NASA, Gene Kranz, described the required attitude for Flight Operations personnel: "...they fully understand that the price of their admission is *Excellence*, and that a Spartan set of standards will govern their conduct. ... Failure does not exist in the lexicon of a flight controller. The universal characteristic of a controller is that he will never give up until he has an answer or another option."⁸

In addition to formal exercises, a significant Disaster Recovery (DR) program was put in place to ensure that in the event of an emergency, such as the ever-present threat of a major earthquake in Southern California, a backup control site could provide a limited Spacecraft Navigation capability. This DR program involved the deployment of several Navigation workstations to the Goldstone Communications Complex (GCC), 200 miles northwest of the Jet Propulsion Lab's main campus in Pasadena. This site, directly tied in to the Deep Space Network's (DSN) array of radio telescopes at Goldstone, would allow Operations staff to receive and uplink commands to the Spacecraft even without the infrastructure support of the main JPL campus. A number of ORT operations were implemented, involving the deployment of Navigation team members for days to the remote site. Due to the limited network communications to Goldstone (aside from the spacecraft communications channels), keeping this site synchronized with the main Navigation computational environment became a real challenge during Launch Operations. (At one point the lead author almost had to be restrained from running "just one more set of backup tapes..." to the very remote site!) Unfortunately, due to funding constraints, support of the DR facility lapsed soon after launch; due to the cost of provisioning personnel, networking, and hardware at the remote site, this effort has not been resumed.

C. Post-Launch to Jupiter Flyby

After the successful launch of the Cassini/Huygens Spacecraft and the initial post launch operations in October of 1997, a second round of system improvements was undertaken. This would provide further good examples for a more formal system design. During this time, the Sun workstations were retired and the HP workstations were upgraded in a scripted manner to HP-UX 10.20. HP-UX 10.20 would require extensive version control as the thousands of patches to the operating system required a clear methodology for configuration management to ensure that all HP workstations would have the same version of HP-UX 10.20 with the same functionality and feature set. Patches from the operating system vendor were assembled into patch clusters that would be applied at one time, to explicitly defined versions of the HP-UX operating system. HP, like several other vendors, would periodically update its operating system release media, which would require the installation of a set of release media with a particular date and part number, along with the assembled patch clusters, to ensure that a given software set containing a given feature set would be installed on the particular machine. This required a much more time consuming operation than in the previous version of the operating system, but the end result was a much more stable and well defined environment for the staff. This would also have the advantage of serving as a useful prototype setup for the types of configuration management problems that would be encountered in working with the open source Linux operating system used later on in the mission.

In addition, the RSYNC⁹ protocol was deployed on the two central file servers so that they could automatically remain in lockstep – the backup server would mirror the primary server every night so that in the event of failure of the primary server, the backup server would contain a day-old copy of everything on the primary server, allowing for improved fault tolerance, albeit with a several minute failover. In addition, this also allowed for a degree of "user

proofing” of the file system. With this backup configuration, if important files or directories were corrupted or deleted by accident, it was no longer necessary to refer to the backup tapes to recover prior versions, possibly days or weeks old. As long as the affected user or users became aware of the problem within the one-day window of the RSYNC cycle, the files could be reverted back to their prior state. (With the commensurate problem that, at midnight, in a Cinderella-like fashion the opportunity for change would be lost!) This was a significant boon to the user community, that, although imperfect, represented a real improvement in system capability at low cost (the largest cost item being the sacrifice of almost half of the disk storage of the Navigation team—a sacrifice that would prove to be invaluable over time).

During this period, it became apparent that the firewall and secure server utilized for the MMNAV Nav Ops Net was in dire need of an overhaul. After extensive effort and numerous staff discussions, a Cisco statefull packet filtering firewall was put in place along with a secure, “bastion host”, mail and SSH server that minimized the number of open ports from thousands to a handful of ports (most of the effort involved went towards convincing staff members that numerous protocols, such as X-Windows and network printing did not need to, and should not, go through the firewall!). The approach used for such a network security upgrade is discussed in Cheswick and Bellovin¹⁰ **, while an excellent roadmap (though somewhat dated) for such an upgrade can be seen in Chapman and Zwicky.¹¹ Although the network would still need significant organizational improvements and hardware upgrades, this significantly improved its front line security defenses.

After the significant difficulties of the upgrade of the operating system, it became clear that some type of system imaging setup that would allow for more automated deployments of Navigation workstations would be necessary. Keeping both development and operational workstations in lock-step configuration management became a significantly difficult and time consuming task. The length of time required to deploy a workstation, and the ease of making a mistake in its deployment (on average each scripted install would take on the order of *ten hours* of careful loading, patching, version checking, securing, and cross-checking) made such a system mandatory. Fortunately HP had included the IGNITE system imaging software set¹² as part of its HP-UX operating system software. Although somewhat tricky to set up, once established, new fully configured workstations could be deployed in under ninety minutes from a master “golden image” stored on the IGNITE server. Patch updates, and ensuring that machines were configured in the same manner became easy, nearly automatic, operations. It became possible to guarantee at a very atomic level that machines had exactly the same operating system software set. This would prove a significant benefit to system administration tasks such as troubleshooting operations problems, as one could ensure that there were no differences in the operating system between two machines, eliminating whole classes of problems.

Furthermore, during this push, it became clear that this imaging not only allowed for clones of workstations to be made in a rapid manner, it allowed for clones of *very tightly secured* machines to be made in a rapid manner. Instead of being a nearly impossible goal, it was now a reasonably easy task to deploy a workstation configured in a highly secure manner. This included locking down file system permissions and running processes and even security features that normally required a high degree of customization, such as secure remote monitoring and process accounting, could be easily configured. This allowed for a significant jump forward in system security, as now all Cassini Navigation machines could be configured in an *identical* and *secure* manner.¹³ This allowed for the formation of not just a “hard on the outside, soft and chewy on the inside...” security configuration that exemplifies most firewalled network security configurations, but true “defense in depth” – a layered approach where multiple systems provide redundant layers of protection.¹⁴ Such an approach (similar in many ways to the multiple-redundant system design approach used as a method to improve reliability) would prove to be an invaluable model for later formal designs.

D. Jupiter Flyby to Saturn Orbit

Another significant improvement to the computational infrastructure occurred during the second half of 2000 that, although successful in greatly improving system reliability and fault troubleshooting and diagnosis, was met with some concern on the part of the MMNAV System and Network administration staff. It was decided to merge

** Although the second edition of *Firewalls and Internet security* is much more up to date, the first edition has a much longer discussion on the mechanics of firewall and “bastion host” construction – this open source approach had largely been supplanted by commercial appliances by the time of the second edition.

network support duties with the younger, but much larger, JPL Flight Operations Network. This process was very successful, for funding was made available to upgrade, reroute, and standardize the network configuration used by Cassini and other MMNAV projects, including the Galileo mission to Jupiter, Stardust, Deep Impact and numerous missions to Mars. Instead of a problematic network design that suffered from years of austere cost constraints, a robust, redundant, modern network infrastructure was put in place. Concerns about this change existed among the MMNAV System and Network administration staff for two major reasons. First, this network would be directly connected to the Flight Operations Network firewall, which, although even newer and more robust than the Nav Ops firewall, had to support far more users and projects (hence open routes in its configuration). Second and more problematic, this network infrastructure would no longer be under the direct support of MMNAV System and Network administrators. We would no longer have control over a very significant part of the critical infrastructure necessary for the MMNAV Navigation Operations network. These concerns would be ameliorated over time due to the sizable advantages present in having network and firewall upgrades supported by another organization, for numerous long needed improvements in performance and reliability, albeit perhaps not in the manner desired, were accomplished through this merger.

In 2002, an effort was put underway to perform a complete upgrade and overhaul of the Cassini Navigation computational environment. Due to the demanding needs of Saturn Orbital Tour Navigation operations, it had been decided to forego this upgrade process as long as possible to take advantage of the ever increasing memory and computational horsepower of current workstations. The oft cited ever increasing computer performance curves of Moore's Law meant that every two years (sometimes incorrectly cited as eighteen months) aggregate computer processor performance would double.^{15,16} The less well known but even more aggressive corollary for disk drive storage, Kryder's Law considers that every year aggregate disk drive storage would double.¹⁷ Requirements for Tour Operations were defined and analyzed over the course of numerous Navigation team meetings, specifying computational, memory and file system storage needs along with reliability and security requirements. A formal process of evaluation and benchmarking of state of the art workstation, server, and file system vendors was implemented in the context of these requirements. These results and associate performance requirements are summarized in the next section.

After significant deliberation, high-end Intel (32 bit x86) workstations and servers, running Red Hat Linux were purchased from Dell Computer Corporation. These workstations would work in conjunction with a Network Attached Storage (NAS) device, purchased from Network Appliance that would serve as the NFS primary file server for the Navigation computational environment. A backup disk array was configured on one of the servers to provide a day-old RSYNC copy of the NAS server (much like the primary and backup file servers did in the Navigation computational environment used after launch). A small number of Sun workstations were also configured into this environment to provide support for Flight Operations ground software components used in other parts of the Cassini Project that would not be ported to the Red Hat Linux operating system. Lifecycle upgrades to the Navigation software and hardware system were planned into the design so that the components of the system, such as Navigation software updates, operating system patches and updates, individual workstation upgrades, and even upgrades to the NAS could be accomplished quickly and efficiently. (For more detail on the configuration of the Tour Navigation computational environment, please see the Appendix.)

This environment, after initial shakedown, would become the Navigation Ground Data System utilized throughout Saturn Tour Operations, significantly outperforming all performance and capability requirements, with almost no system-wide downtime, during an aggressive orbital tour of the Saturnian system including the Launch and Landing of the Huygens probe, fifty-two flybys of the largest moons of Saturn, and more than one hundred and fifty precision orbital course corrections through the end of prime mission on June 30, 2008.

III. Requirements from Navigation Tour Specifications

The successful design of the Tour computational environment for the Cassini Navigation team involved the analysis of performance and capability requirements gathered from the Navigation team and of requirements on reliability, availability, and security derived from several years of successful operations and lessons learned along the way. Many of the Navigation team performance requirements used launch and cruise operations performance as a baseline for the requirements of Saturn Tour operations.

Navigation team performance requirements for Tour Operations derived from the concern that the computer hardware used in the Navigation computational environment during launch and cruise operations did not have the computational horsepower to successfully complete the Saturn Orbital Tour within acceptable limits. Initially, these requirements were defined in terms of processing capability and speed of key software sets used by the Trajectory Analysis, Optical Navigation, Orbit Determination, and Maneuver Analysis components of the Cassini Navigation team. They included requirements to update the satellite ephemeris of the nine major Saturnian satellites in a single run (necessitating temporary file space requirements exceeding 1 GB!), requirements on Orbit Determination software for parameter fields and data sets expected during Saturn Tour, Trajectory Analysis requirements based on the scale of Saturnian operations, Optical Navigation requirements covering Optical Navigation picture processing, scheduling, storage requirements, and Maneuver Analysis design requirements for Tour Operations. Unfortunately, most of these requirements were based around models and software sets which did not exist when these formal design parameters were being codified. (Several software sets for Saturn Tour were not implemented due to the fact that the computational environment itself had not been chosen, a difficult “chicken and egg” problem!) In order to resolve these difficulties, a set of standard requirements was chosen that would encapsulate specific Tour requirements by performance requirements for the entire Navigation computer environment that would meet or exceed the individual Tour software requirements. They stated key requirements:

4.18 The Navigation Hardware and Operating System Software shall be benchmarked in its current operational [Launch/Cruise] state, on both client and server systems, using evaluation tools including industry standard CPU benchmarks as well as NBODY (Section 312) suite of benchmarks. These benchmarks should be used to determine and obtain hardware that at a base level is five times (5x) faster than the current Navigation Hardware [ten times (10x) goal] for Tour operations.¹⁸

From these requirements, the Navigation software benchmark utility NBODY (an in-house hardware benchmarking tool, using Navigation algorithms built in C, Fortran 77 and Fortran 90) was utilized along with the results from the industry standard SPEC2000 CPU¹⁹ set of benchmarks to derive benchmarks for the Navigation hardware used at that time for cruise operations, and then to benchmark state-of-the-art workstation and server models that represented leading candidates for upgrades to the Navigation computational system. Results from cruise hardware served as a useful baseline to evaluate then-current vendor offerings for workstation systems:²⁰

Table 1. Cruise hardware specification: NBODY results

Cruise hardware specification under HP-UX 10.20, NBODY results, (lower is better):		
Type of Benchmark:	HP J2240 (Server), 240 Mhz, 1.5 GB RAM, HP 10.20:	HP J210XC (Workstation), 120 MHz, 512 MB RAM, HP 10.20:
NBODYC:	8.3s	23.3s
NBODYF77:	5.1s	18.8s
NBODYF90:	5.1s	21.8s

(These metrics use the same NBODY algorithm compiled with C, Fortran 77, and Fortran 90.)

Table 2. Cruise hardware specification: SPEC2000 CPU results

Cruise hardware specification under HP-UX 10.20, SPEC2000 CPU (Est.), (higher is better)		
Type of Benchmark:	HP J2240 (Server), 240 MHz, 1.5 GB RAM, HP 10.20:	HP J210XC (Workstation), 120 MHz, 512 MB RAM, HP 10.20:
INT:	176.0	62.6
FP:	159.0	59.0
INT_RATE:	4.02	0.72
FP_RATE:	2.71	0.67

(In the SPEC2000 CPU benchmark scheme four separate categories are considered: maximum integer performance [INT], maximum floating point performance [FP], integer processing throughput [INT_RATE], and floating point processing throughput [FP_RATE].)¹⁹ With these performance baselines, selected vendor workstations could be evaluated against these software metrics:

Table 3. Selected vendor hardware platforms: NBODY results

Performance Benchmarks on selected vendor hardware platforms: NBODY results (lower is better):				
Type of Benchmark:	Sun Blade 2000, 1.05 GHz, 1 GB RAM, Solaris 8:	x86 PC Workstation, 3.0GHz, 1 GB RAM, Red Hat Linux 7.3:	HP Itanium 2, 1.0 GHz, 1 GB RAM, Red Hat Linux 7.3:	HP C3700 750 MHz, 1 GB RAM, HP 11.0:
NBODYC	1.72s	0.86s	1.48s	2.30s
NBODYF77	1.72s	0.96s	0.73s	2.40s
NBODYF90	1.72s	0.92s	0.73s	1.80s

Table 4. Selected vendor hardware platforms: SPEC2000 CPU results

Performance Benchmarks on selected vendor hardware platforms: SPEC2000 CPU results (higher is better):				
Type of Benchmark:	Sun Blade 2000, 1.05 GHz, 1 GB RAM, Solaris 8:	x86 PC Workstation, 3.0GHz, 1 GB RAM, Red Hat Linux 7.3:	HP Itanium 2, 1.0 GHz, 1 GB RAM, Red Hat Linux 7.3:	HP C3700 750 MHz, 1 GB RAM, HP 11.0:
INT:	537-610	1060-1103	807	568-604
FP:	701-827	1025-1115	1356	526-576
INT_RATE:	6.2-7.0	12.3-12.8	9.39	6.59-7.00
FP_RATE:	7.3-8.4	11.9-12.0	13.6	6.10-6.68

Although useful, this CPU performance characterization would represent only part of the requirements derived from the Navigation team. Capacity planning, in terms of disk storage capability, and memory requirements for associated workstations would also be evaluated. As the requirements were being considered, some discussion as to the overall architecture of this environment became an important part of these considerations. Possibilities were entertained for a number of different configurations: the classic “star” configuration utilized during Launch/Cruise operations, a purely peer-to-peer approach (an improved version of the cross-mounted NFS directory approach considered before launch), or a “thin client” approach where all processing would take place on a few high end server class systems. Examining a number of concerns helped us to evaluate these differing architecture models.

Possible Navigation Architecture Design Configuration:

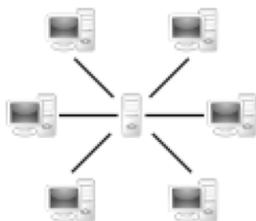


Figure 2.
Classic “star” configuration

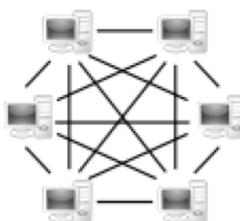


Figure 3.
Peer-to-Peer configuration

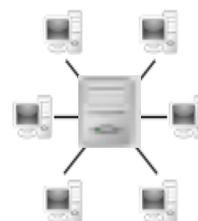


Figure 4.
“thin client” configuration

The peer-to-peer approach, with each node connected to every other node as seen in Fig. 3,²¹ was rejected due to the high value in having one set of software distributed to all workstations through a network file system (no central servers). Although tools such as RSYNC, would allow for a peer-to-peer network to be Configuration Managed effectively, it was clear that there would still be a need for some centralized services, running on one or more central servers, which would not function in a peer-to-peer model. These included tools that would gather and distribute various spacecraft data sets and software packages that would help automate routine tasks in the Maneuver Analysis function. With such needs, it was not clear what advantage the peer-to-peer approach would possess, and it clearly suffered from a much higher degree of complexity than other architecture approaches.

The “thin client” approach, in the classic “star” configuration with the large server as seen in Fig. 4,²² had a number of advantages in its design that made it an attractive option. Instead of the expense of purchasing systems for Navigation engineers that could each individually meet the processing power requirements for Saturn Tour Operations, a few high-end servers would be purchased and utilized by the whole team, while individual Navigation engineer workstations would serve only as “dumb terminals” (or thin clients). This option represented an effective choice; however, a number of factors rendered this a poor option for Navigation architecture. The first concern involved the computationally expensive nature of Navigation Operations software. Unlike most software sets that are a good fit for such centralized computation, Navigation software’s numerical processing used aggressive amounts of memory, CPU power, and disk space during its operation. Moreover, it suffered from performance problems and bugs that could cause such resource utilization to spiral out of control. Furthermore, utilization of the software was difficult to control with traditional resource allocation techniques such as disk quotas and memory usage limits, as almost all such approaches relied upon oversubscription of such limited resources to be effective. Like the hotel, restaurant, and travel industry, many system resource controls rely on the premise that most users will not use all (or, on most computer systems, even a significant proportion) of their allotted quota of resources. Navigation software had the hallmark of using all available system resources in its computation cycle; indeed, some Navigation software was notorious for bringing the largest and fastest machines available to their knees in the event of a bug or even poorly chosen inputs to the software! This is a user community that by its fundamental nature has difficulty sharing resources. Finally, the issue of cost became the definitive concern in that the high cost of central servers capable of supporting Navigation processing requirements was significantly greater than the cost of individual high end workstations that could meet such performance requirements.

From these architecture considerations, the classic “star” approach, as seen in Fig. 2,¹⁵ provided a middle ground that was a best fit for Tour operations, offering the benefits of ease of Configuration Management of centralized file service and processes needing centralized services, while allowing for numerical computation to be performed on the end node workstations rather than on the central server. The method of resource allocation would be that of “one engineer, one workstation” for normal operations, with overflow capability in the system, so that each engineer could perform all computational tasks on their individual machine (without concern that they would encounter

interference from other engineers). With this architecture approach having been decided, a number of performance and capability requirements could be codified for Tour operations:¹⁸

- All Navigation Engineers would have their own workstation in the Navigation Operations area.
- Each workstation and server would be capable of meeting the performance functional requirements for Navigation Operations.
- Each workstation and server would be connected to the MMNAV Nav Ops Net (avoiding communication problems with remote machines, perhaps external to the Flight Operations Firewall).
- The MMNAV Nav Ops Net would be capable of transferring data at Gigabit speeds (needed for the file size and file system loading expected during Tour Operations).
- Each Workstation and Server would have the storage capability for 150 GB of local file system storage (mirrored).
- The Navigation computational environment would support online data storage for all navigation delivery files, and files necessary to duplicate such deliveries, until the prime end of mission (estimated at 3 TB). The Navigation computational environment would be configured so as to allow online disk storage to be easily scaled up to five times its capacity to provide for future growth.

These requirements were combined with Quality of Service (QoS) and Mean Time to Restore (MTTR) specifications on the system:¹⁸

4.8 The Navigation Hardware and Operating System Software shall provide 99.97% [i.e. 2-3 hours unplanned downtime per year] uptime capability.

4.9 The Navigation Computer System shall be configured to have a mean time to restore overall system functionality of 30 minutes during critical periods and 60 minutes during non-critical periods. While this does not imply that all subsystems will be functional, all systems necessary to fulfill the NAV operational requirements will be restored in this period.

4.11 The Navigation Hardware and Operating System Software shall provide 24-7 uptime capability.

IV. General Design Principles

With these performance, capability, and reliability requirements as well as lessons learned from Launch and Cruise operations, a framework emerged for a system design. However, several general principles would also be incorporated into the Navigation Ground Data System Design to provide a successful computational system for Navigation operations during Tour. Requirements for system fault tolerance, security, and configuration management came from successful operations and feature improvements to operations during interplanetary cruise. Although the performance requirements drove many aspects of the workstation, server and networking benchmarking and design, these more general requirements derived from lessons learned along the way will have more long term utility as an example to other efforts. With these principles, not only would the requirements for the system be met reliably and efficiently, but troubleshooting and system repair, system installation and reconfiguration, software installation and regular maintenance would be greatly simplified.

A. Configuration Management

The first of these general principles, Configuration Management, would have several large improvements over the Launch and Cruise configuration. Similar to the effort set up to support the numerous software revisions and patch kits utilized for HP-UX 10.20 during the spacecraft cruise, a regimented system of software installs and

patching was put into place to support the Red Hat distribution of Linux. A specific set of distribution media (determined by manufacture date) was chosen and loaded onto a x86 PC workstation. A carefully scripted set of software was installed from the distribution media, and then specific patches were applied against the machine. Great care was taken to get accurate version numbers of package sets on the machine (called RPM's on Red Hat Linux for Red Hat Package Manager) and then replicate that installation on another machine. These installs were done not only on carefully selected hardware platforms that were known to support the Red Hat distribution of Linux, but also on hardware platforms that were carefully chosen to be as similar as possible to each other (difficult indeed on the PC platform where part and even hardware component vendors can and were swapped out almost at will by PC manufacturers to get the lowest possible component price). This was accomplished by defining precisely the exact configuration desired as part of the initial purchasing, and whenever possible, buying in bulk (really the only effective approach, as hardware bought at exactly the same time would often use the exact same part configuration).

We used a software tool to automate a large portion of this effort. Much like the IGNITE tool used on HP-UX, an open source tool known as SYSTEMIMAGER²³ performed automated installations of machines. Now the laborious scripted software installs would only have to be performed on one machine and then the precise configuration could be copied over to an install server. Once the server was set up, it was capable of performing "bare-metal" (i.e. on a workstation without any underlying operating system installed) installations of these customized operating systems in an average of fifteen minutes. By design, all non-ephemeral user files would be located on the central NAS NFS server, so these machines were configured to be clones of one another with file systems that should be static (except for swap and temporary files used by the users). (As noted, this is a similar approach to the efforts first seen in the Andrew Distributed Computing Environment at Carnegie-Mellon.)⁶ With this system in place, it became possible to completely re-install a customized version of the operating system on nearly thirty Cassini Navigation workstations and servers in about eight hours. These machines would, down to a very atomic level, have an *identical* set of files on them, which would not only allow for very hardened security configurations requiring considerable customization to be configured, but such installations could be cryptographically validated, file by file, against the original "golden image" stored on the SYSTEMIMAGER server. Indeed, even on machines with differing hardware configurations, all that would need to be changed would be the configuration files and file system links to point to the appropriate drivers for a given machine architecture. If a machine was modified, either by accident or by system compromise, the changes made to the file system on that machine could be determined and, if desired, corrected in a manner of minutes. Troubleshooting and repair of problems became far simpler as it became clear that if a given problem occurred on one machine, and not on another *identical* machine, then the problem must be caused by a user error or a hardware error. (Indeed, in one case, this increase in troubleshooting capability and sensitivity enabled us to determine that a specific software routine used by a Navigation program was in fact sensitive to the slight differences in the Arithmetic Logic Unit (ALU) used on the CPU between two x86 processors!)

Further aiding the CM task, the central file server chosen, a Network Appliance NAS NFS file server, had a number of features that helped improve the server-side of CM. In addition to the previously noted advantages of using one centralized copy of a given file or software set, large amounts of disk were set aside for storage of Navigation Flight Software. This would prove to be an exceptionally effective use of resources! It enabled the storage of multiple versions of the software used on the Navigation computational environment. Now instead of having to delete older versions of software, as had to be done to save space on the previous configuration of the Cassini Navigation computational system, it was possible to maintain many different versions of software. With carefully set up file system links and well configured software execution "PATH" statements, it became possible to select any prior combination of software quickly. Although a well defined default setup used normally by the system users would always point to a current, Configuration Managed software set. In addition, the Network Appliance file server had a feature known as "snapshotting"²⁴ that would, once configured, make a backup image of the file system every few hours that would be archived for up to one month. Now both users and system administrators could revert to numerous prior versions to recover a file, directory, or even a whole previous configuration of the file system. As with the previous day-old RSYNC⁹ server "user proofing" of the file system, if important files or directories were corrupted or deleted by accident, it was no longer necessary to refer to the backup tapes to recover prior versions. In this setup however, instead of just maintaining a day-old copy, multiple versions would be maintained, going back one month. This setup was further improved over the previous cruise configuration in that the Network Appliance

file system used clever mechanisms in the file system designed to implement these “snapshots” so that they would only take up a fraction of the disk space used by the RSYNC server. Indeed, instead of a full copy of the file system, taking up twice the total space of the single copy, only *the changes* to the file system between the older and newer versions of a file were saved. An analysis of overhead on the Navigation computational environment shows that on quiet file systems with few changes these “snapshots” could occupy less than 1% of the total file system space used.

B. Fault Tolerance

The second of these general principles, that of integrating fault tolerance into the underlying system design, would also be greatly improved over the Launch and Cruise implementations of the Navigation computational environment. Instead of having to add fault tolerance to an already established design, fault tolerance concerns were considered as part of the initial specifications and purchasing decisions. These fault-tolerance principles which the system design was based around included *modularity*, where the system architecture is divided into separate modules that, upon failure, can be replaced with a new module, *fail-fast*, where a system component will either work correctly or stop immediately, *independent* failure modes, in that if one module fails it does not affect the other modules in the system, and *redundancy and repair*, where spare modules are configured so that when one module fails a second can replace it almost instantly, while the first can be repaired or replaced off-line.²⁵

Each of the PC workstations was identically configured. Superior quality and more reliable components (such as SCSI or SAS drives in preference to IDE/SATA drives) as well as spare parts were purchased for components likely to fail, such as disk drives. As the requirements for Quality of Service for MTTR were written from the view of the whole Navigation computational system, the individual PC workstations were seen as functional units—if one failed, it was considered, by design, to be easily replaceable with another unit, as each was a clone of the master “golden image.” Furthermore, almost like in a RAID array, a spare office was set up with a spare workstation, so that in the event of a failure of an individual workstation, users could simply go to the spare office and continue on an identical workstation until the workstation in their own office was repaired or replaced. Like the “snapshots” discussed previously, which also served to promote improved fault tolerance on the part of the users of the system, these spare machines and office setup enabled the disentanglement of system failure from both a system administrator response perspective and an event which prevents users from completing their tasks. Now a user could simply move their work to another workspace and continue working, while the ever-busy system administrator could fix the machine at their leisure. (It should be noted that in practice it was not quite so simple, as users would normally rather wait to have their own machine fixed rather than move their work to a separate office. However, it did serve as a gauge to the criticality of the user’s needs: if they *needed* to finish a task, they would use the spare office!)

Moreover, in terms of the server and peripheral systems such as the printers, similar approaches to fault tolerance would also prove to have great utility. We purchased three identical x86 PC server class machines in a dual processor configuration with twice the memory of the user’s workstations. These machines were configured with the same image as the user workstation, once again aiding CM, with one server that was designated as the “prime” server that would run processes that needed to run on a central server for the entire Cassini Navigation Ground Data system. This includes services such as processes to gather and store, or even broadcast locally, certain spacecraft data sets taken from real time and near-real time downlinked data from the spacecraft. This also comprised centralized tools that aided in automating certain Navigation processes such as the Maneuver Automation software set. The other two server machines would be configured to take over the role of the “prime” server in the event of a failure. Users were trained to switch some of the critical user processes, where possible, from the “prime” server to the “alternate” server in the event of a failure. In addition, peripheral systems were configured in a similarly redundant manner. The Sun workstations that had been purchased to support Cassini Project Flight Operations ground software (the majority of Flight Operations workstations used at JPL are currently from Sun Microsystems) that could not be ported over to the Linux platform were, like the Linux PC workstations, cloned, and a spare Sun workstation was placed in the spare office. In like manner, a primary network printer was set up for the workstation users, as well as several redundant backup network printers.

Behold, the fool saith, "Put not all thine eggs in the one basket" – which is but a manner of saying "Scatter your money and your attention"; but the wise man saith, "Put all your eggs in the one basket and-watch that basket!"
-Puddin'head Wilson's Calendar, as quoted in *Firewalls and Internet Security*²⁶

In the prior systems noted, fault tolerance was improved, much in the manner of a RAID disk array, by adding additional functional units to a resource collective or pool to increase reliability, by ameliorating failure cases. However, some systems in the Navigation computational environment could not be improved in this manner and so, as seen in the quote above, other approaches to improving fault tolerance needed to be considered. Functionally, the most significant point of failure in the Navigation computational environment was the central NFS file server. Indeed, this was one of the significant drawbacks to the classic "star" architecture: the central server was a critical component without which the rest of the configuration would probably not function. There were several ways to mitigate this risk, ranging from making changes to the client end nodes (the workstations and servers in the Cassini Navigation example) that would permit the clients to survive for a length of time without the central server (which however would move the architecture design to a more peer-to-peer configuration that could play havoc with CM) to improvements to the central server. As considered in the quote above, the second option was taken in the Navigation environment. Due to the complexity of the central file server, obtaining meaningful reliability metrics would prove impractical, however a similar setup by Santonja, Molero, Alonso, Serrano, and Gil, utilizing a RAID-5, configuration, although not an identical configuration, would provide guidance and approximation of the reliability improvement achieved by differing components of such a multiple-redundant setup (they consider not only the usual concerns about disk failure, but also evaluate failure rates of support equipment such as controllers, cabling, power supplies, and even the time it takes for spare disks to be shipped from the manufacturer and replace failed units).²⁷ As noted before, to fulfill these high availability requirements, a high-end Network Appliance NAS NFS server was purchased. This multiple-redundant system had dual redundant power supplies (tied into different electrical circuits) on each of its subcomponent systems (network interface head and multiple disk trays) with three multiple redundant network cards (in three different ports) configured as a hot-hot-warm backup of one another (in the event of a failure of one of the network ports one of the other two continue to serve data at a degraded capability). This system served data from several disk shelves containing more than fifty fiber channel disks, with dual redundant connections to dual redundant controllers, configured as part of a modified RAID-4 disk array. Every component, except the whole system itself, had at least dual, or n+1 redundant components. This system had mechanisms for automatically (without human intervention) swapping failed components (such as disks) out with working components from a spare pool. To improve further the robustness of the Navigation environment, a RAID array from a different manufacturer (deliberately, to avoid possible unknown faults with Network Appliance) was attached to one of the spare servers that behaved in the same manner as in the Launch and Cruise configuration, in that RSYNC was used to backup the central file server every night.

From our considerations of fault tolerance we have had to utilize empirical approaches based on prior experience to achieve our requirements for Quality of Service and MTTR. Although approaches for improvements to reliability are clearly understood, it can be difficult even under simple, limited cases, without artificial constraints on the system, to analyze system dependability and derive quantifiable metrics for system evaluation.²⁸ This system design sought to minimize single points of failure and ameliorate those points that could not be eliminated. The approach is to do all that is possible within the budgetary and time constraints available, and then test and examine the empirical results to see if they meet the established QoS and MTTR metrics requirements. If the system, by trial and error, does not perform up to the required values, parameters are examined and modified to bring the system within required values. Indeed, given enough resources, how far one chooses to go to evaluate and then improve system reliability in such efforts are almost always due to the limits of time and money! To illustrate this point, consider the far end of the spectrum, where QoS requirements would be very high and there would be virtually no limit on resources (with virtually unlimited budget and schedule freedom). Yeh provides us with such an example when he describes the flight avionics systems, such as the Primary Flight Control System found on the Boeing 777, that achieve ultra high reliability metrics and had enormous design resources in terms of budget and time. The system portrays a highly available, highly redundant, system used by the fly-by-wire avionics controls, that is a Triple Modular Redundant (TMR) system (usually the highest order of redundant design with three redundant components for each single point of failure) providing QoS metrics (required for a commercial aircraft) specifying a rate of failure of the flight controls of less than 1.0×10^{-11} per flight hour. To obtain these failure metrics required programs

of trade-off design study, analysis and simulation of failure modes, 20,000 lab-hours, and 3,800 flight-hours of flight testing as part of the evaluation process!²⁹

C. Security

The problem becomes far more acute when one considers the possibility, not of random chance, but of intelligent actors as a failure mode. Considering security as a measure of reliability does provide the benefit that systems that are hardened to be fault-tolerant against intelligent actors will often prove robust against numerous “natural” failures as well. Security design takes on several principles that have similarities to architecture in that the strength of a system can often be improved, not by what one adds, but by what one takes away. As noted above, secure systems are considered, not as a single defensive stronghold or chokepoint, but as a series of overlapping defenses, one after the other, much like other fault-tolerant multiple-redundant systems, characterized as “defense in depth.”²⁶ This approach is based on the idea of likening security to a series of overlapping walls, much like World War I trench warfare or a Medieval castle-in that one presents so many barriers to entry that a hostile adversary will choose to pick a less well defended target. Cheswick, Bellovin, and Rubin characterize this approach in that “...we use a restricted meaning for the word ‘secure’ when applied to a host. There is no such thing as absolute security. Whether a host is penetrated depends on the time, money, and risk an attacker is willing to spend, compared to the time, money and diligence we are willing to commit to defending the host.”³⁰

The security design of the Navigation computational environment is based around the cornerstone security principles of Confidentiality, Integrity, and Availability (CIA). These principles define the key concerns in securing a system, not in terms of specific technique or subsystem protected, but in terms of what things about the computational environment need protecting (i.e., if one considers a bank vault’s security, the concern was for what needs protection-the contents of the bank vault-not the design of the bank vault). Confidentiality concerns the concealment of resources or information, much like the oft-repeated military principle of “need to know,” Integrity is concerned with preventing unauthorized changes to the system, and Availability is the concern that an unavailable system can be as bad or worse than no system at all.³¹

For the Navigation computational environment these traits manifest in a different manner than as seen in military or financial systems. While Confidentiality is extremely critical in military computational systems (in some cases destruction of the system is more desirable than the unauthorized release of information), and of high priority for financial systems (where customer data is not only a crucial part of business operations, but strong legal regulations also come into play for control of consumer information), it is less critical in the context of the Navigation computational environment. Confidential information for the Navigation computational environment includes the (very important) password databases and password authentication system, used for system access as well as detailed information on system and network configuration (which could be used by an intelligent adversary to subvert security). As an example of such information and the importance of Confidentiality as a baseline configuration, the naming of individual nodes can be seen to be an important part of Confidentiality – because host names may reveal a great deal of information about the underlying network design.³² For example, consider the hostname “casnfs3,” which relays the immediate information that the machine is a Cassini server, running NFS, and that there are at least two other Cassini NFS servers. This is a strong argument for individualized but non-informative host names.^{§§} Another important part of Confidentiality is concerned with the control of information either critical to mission operations or which has not been cleared for public release (although we are a public civilian mission, access to some data sets are restricted).

Integrity continues to be a very crucial trait in the Navigational computational environment, as unauthorized or improper modification of the system could lead to very serious problems. Integrity can be seen as being broken down into two separate subcategories, that of prevention mechanisms (which comprises the gamut of the security configuration of a system) serving to deter unwanted intrusion by means of “defense in depth,” and detection mechanisms, in which after-the-fact systems determine when system integrity has been compromised. Detection

^{§§} The lead author can recommend that a number of Science Fiction and Fantasy series provide a wealth of objects and characters that serve as effective host names. In addition, D. Libes’s “RFC 1178 –choosing a name for your computer” can serve as excellent guidance.³²

systems on the Navigation system include tools such as programs that monitor logs from the Navigation computer systems. Other tools served to validate the integrity of a file system cryptographically, looking to observe what changes have occurred since a last known “good” configuration, such as the SYSTEMIMAGER installer mentioned above, and the TRIPWIRE³³ cryptographic file system integrity checker. Finally, a locally developed real time log monitoring system provides immediate detection of anomalous system faults (whether hardware and operating system related or caused by unwanted intrusion).

Availability in the Navigation computational system, coupled with the reliability constraints from concerns about fault tolerance, had the additional concern that someone might deliberately try to deny access to data sets or a system service. The mechanisms used to improve reliability, such as redundant units, resilient components, and multiple-redundant systems, provided some protection against attempts to subvert availability, but further protection would require additional changes to the system design. Operating system configurations for systems such as networking and network file services were radically modified to increase memory pools, counters, and communication timing in an effort to prevent attempts to launch Denial of Service (DOS) attacks against the Navigation computational environment. In addition, many unsafe toolsets were either removed or stripped of the privileges necessary to be used in a hostile manner on the Navigation computational environment. Furthermore, many files and directories (especially system files and directories) were restricted to a “read-only” state, to even further defend against unwanted modification, so that even if flaws were found and utilized in the security setup, a penetration would not provide sufficient access to the system to make changes that would affect system availability.

While the concepts in CIA capture useful concerns regarding what was in need of protection in the Navigational computational environment, the opposite side of such concerns, considering security as an aspect of reliability, would prove useful to encapsulate these concerns in an approach for general hardening of the system against external intelligent actors (i.e. going back to the example used before, this focus is on how the bank vault will protect its contents). Cheswick, Bellovin and Rubin³⁴ offer an excellent summation of the process of hardening a host against outside attack for web servers and other Internet information servers, to the point where they render it very difficult to break into the machine without direct physical access. This level of concern and effort is similar to the design goals for the Navigation computational environment. They remark, “It is not that difficult to make a specific host highly resistant to anonymous attack from the Internet, The trick is to have that host remain useful.”³⁵ Likewise the Navigation computational environment is designed with “defense in depth” as a paradigm, in that should an attacker get past the Firewall, they will encounter a very hostile network of layered defenses to make any further progress futile. As noted by Cheswick, Bellovin, and Rubin, the difficulty comes in securing such a system while keeping it functional for legitimate users. This process of external hardening, as noted above, had principles with similarities to architecture in that the strength of a system can often times be improved, not by what one adds, but by what one takes away. Very secure systems can be constructed by minimizing a system configuration, but such systems are not generally appropriate for a “normal” user population. (E.g. the removal of the X-Windows system and associated window managers can be clearly seen to improve the security posture of a machine, but few users would desire to interact with a local machine over a terminal connection.) In the Navigation computational environment, the system hardening process involved the characterization of what services and activities the local user population either needed to complete their Navigation Operation critical tasks (e.g. the successful function of mission critical Flight Software and delivery of Operations data sets to appropriate parties), or what the user population thought was necessary for completion of their job role (e.g. functional compilers, formatting and visualization tools, editors, viewers and third party software sets such as MATLAB), or what they believed the system would not be useful without such a capability (CD and DVD writing capability, and the ability to mount CDs and DVDs in a secure manner without system administrative assistance, as well as X-Windows). With this characterization, a clear delineation between what actions a user would be expected to perform and what actions a user would never be expected to perform was drawn. From this characterization, software sets and privileges a user was never expected to need were removed from the system image, and restrictions were put in place so that users would have difficulty in performing anomalous actions. As noted above, a preliminary effort during Cruise served as a useful prototype for this system hardening activity.¹³

From this prototype, seven steps could be categorized as a part of the system hardening process for these Linux systems after installation and patching:

1. *Shutdown of all unnecessary network services*
This includes any services supported by INETD or XINETD – there should be no services that INETD/XINETD should support on a secure system (while there may be some reason to use services such as ftp, finger or one of the other “legacy” services in INETD/XINETD, there is no call for the server daemons to be enabled on a secure machine). With some effort it became possible to strip the system down to where only SSH and a handful of other services were running.
2. *Configuration of the network stack to be robust against numerous Denial of Service (DOS) attacks*
This was accomplished by modifying the kernel network configuration files such as “/etc/sysctl.conf” to set default settings to much higher values to disable the resource exhaustion techniques utilized by numerous DOS attacks.
3. *Shutdown of all unnecessary system services*
As was done with the network services in the first stage, all unnecessary system services were shut off.
4. *Local and network file systems were reviewed and restrictive permissions were set up*
File systems such as “/usr/local” had the most restrictive permissions possible, such as read-only access. Removable media (CD/DVD/USB drives) mounts were set up so that they would not support devices or setUID and setGID programs (which could be used to circumvent system security and give the user elevated privileges).
5. *The file system was scanned, looking for files and directories that had poor permission settings*
World writable files and directories were minimized. The set of setUID and setGID executables was minimized to a very limited and well-defined list.
6. *Logging was set up*
The system was set up to save log files, and, additionally, changes to the system were made to transfer them to a remote log server for storage and analysis.
7. *Finally, the seventh stage involved the deployment of a customized host-based firewall*
This further restricted access to only a few limited TCP/IP TCP ports. The machine could make a few connections, such as SSH, but otherwise would not respond to any network traffic, including the venerable network tool PING.

These machines were still useful to the users of the Navigation computational environment, but both internally and externally they had been hardened against system compromise by hostile adversaries. Once this effort was completed, the system was evaluated and fingerprinted using tools such as TRIPWIRE and then the SYSTEMIMAGER software was used to take a “golden image” of the machine and load the image on the install server for further use.

A mechanism for deriving metrics to evaluate the security level of a machine was a necessary part of this effort. Fortunately, a number of tools exist to evaluate a machine’s internal and external defenses. A number of tools were selected to assist in this effort, but other tools may make more sense in other environments. To conduct external network scans of the machine, industry standard network scanning tools such as ISS (Internet Security Systems)^{36,38} and NESSUS^{37,38} determined the open ports on the system, with the custom firewall in an up and down configuration (to examine differing layers of the network-based defenses). Using differing approaches, each of these tools determined what ports on the system would respond, and then would attempt to determine if the software listening on these ports was vulnerable to a catalog of hacking techniques. After these results were generated any problematic results were corrected and the scans run again until a system with a minimum set of open external holes was achieved.

In a similar manner, an internal host scanner from the Center for Internet Security (CIS)³⁹ was chosen to perform scans of the internal defenses of the machine. CISscan had three significant advantages which made it a clear choice for this effort. First, it had a consistent set of metrics for evaluating host security on a variety of operating system

platforms, which enabled comparisons between differing machines, to be made. Second, CISscan represented an effort to develop an industry standard “best practices” series of guidelines for system security based around a consistent set of metrics, with the CISscan tool used to check a given system against such guidelines. Third, it was also very well documented, so that the security implications of each tested system component was well understood and could be evaluated against the requirements of the system. With this too, although its metrics were somewhat subjective, it was possible to establish a baseline security level, and ensure that patching and operating system changes continued to meet those metrics.

D. Human Factors

From these considerations of Security, Fault-Tolerance and Configuration Management a more general principle was derived that also could be applied to the support of this well configured system. Using ideas gathered from Configuration Management, scheduling and implementation of software patching, operating system updates and hardware upgrades was tightly managed. During Saturn Tour Operations, the system was expected, as noted above, to function 24x7, with less than 2-3 hours of unplanned downtime per year. While it was possible to bring down redundant components of the system, such as individual servers, workstations, or printers, the whole system was not to be brought down except under the most extreme cases. Even bringing down redundant components had to be scheduled for time periods where no significant Spacecraft Navigation Operations would be underway, as the continued smooth functioning of the Navigation computational environment was considered crucial to mission success. These time periods of critical activity and then relative calm were determined by the Spacecraft mission sequence and the motions of Saturn, Titan, and the numerous smaller moons and moonlets of the Saturnian system. The times of peak activity and calm were not tied into the 9-5 regular work day schedule or even the Monday through Friday work week schedule. Sometimes it was possible to perform major system maintenance on a Monday afternoon, while the previous weekend had seen many Navigation engineers working until the early morning hours of Sunday to handle a particular critical event. (Indeed it was a bit of a running joke that one of the unwritten requirements of the mission, as implemented by a few particular scheduling planners, was to try and eliminate all of the JPL scheduled holidays!)

Maintenance and emergency repair operations were designed, as noted above, with fault tolerance as a guiding design principle. Another key consideration, observed from Launch and Cruise operations, was ease of restoration in the event of a critical situation. As system administrators are well familiar, it is in a crisis one appreciates a well designed recovery procedure, and conversely is the worst time to deal with a confusing, overly complicated setup. This design paradigm was a central component in the design of numerous components of the computational system. As noted in the discussion on fault tolerance, a spare office and several spare components, including a server and disk array comprising a day-old copy of the entire primary file server were configured as hot backups of the primary computational environment. In the event of emergency, the backup server with the disk array attached could be moved to a different location and function as a stand-alone copy of the *entire* Navigation computational environment. During the failure of a user workstation, users should be able to resume work using the spare computer systems without even having to involve the system administrators. Under such a setup the restoration of a functional configuration of the Navigational computer system should be possible within the Quality of Service MTTR requirements in all but the most catastrophic failure cases. In the event of an emergency where the primary fileserver would have to be restored from backup tapes (consider just how many components would have to fail in a line to necessitate such an event!), such tapes were comprised of clearly labeled “dump archive format” tapes representing level-0 backups of the file system. In other words, the backup tapes are configured in a format that can be read on almost any Unix or Linux system, and that only one tape, or sequence of tapes, would have to be read for a full recovery of a file system. Most system components are interchangeable, and even if the spare parts supply was exhausted, functional workstations could be scavenged from spare workstations. Moreover, with the SYSTEMIMAGER server in use, in the event of a serious configuration problem on a workstation or server, a “known good” copy could simply be re-installed from the server in a manner of minutes—often faster than the traditional troubleshooting and repair process would take. With this approach of including the system administrator as part of the recovery process, and considering the means to make the recovery process as easy and as rapid as possible, the system administration staff became a central part of the QoS guarantee for the system.

As a part of the support of the Navigation computational environment, the system administrative staff had to not only plan maintenance operations on the Navigation system, they also had to be scheduled for on-call and on-site support of the Navigation computer systems. Although almost all the time this was an uneventful, “Maytag repairman”⁴⁰, activity (which was the strongly desired goal for such operations), it was deemed critical that system administration staff be close at hand to resolve any problems that might arise as part of the Navigation data processing. During the busiest parts of the mission, these activities grew to nearly a seven day a week, sixteen (or greater) hour a day schedule supported by two full time system administrators. The difficulties in scheduling such round-the-clock support were severe – as this was a long term mission with a four year primary tour, the usual strategies of a shorter length mission would not work. With no end in site, burnout could be a real problem. A scheduling paradigm would have to be created to help maintain the health and sanity of the system administration staff, even during the most demanding parts of the mission. Several scheduling approaches were put in place to ameliorate the worst difficulties of the schedule. As an example, consider Fig. 5, describing one of the busier two week periods in the mission schedule, supported by the lead author, Frank Yu, and Scott Fullner-a Navigation system administrator supporting another operations team:

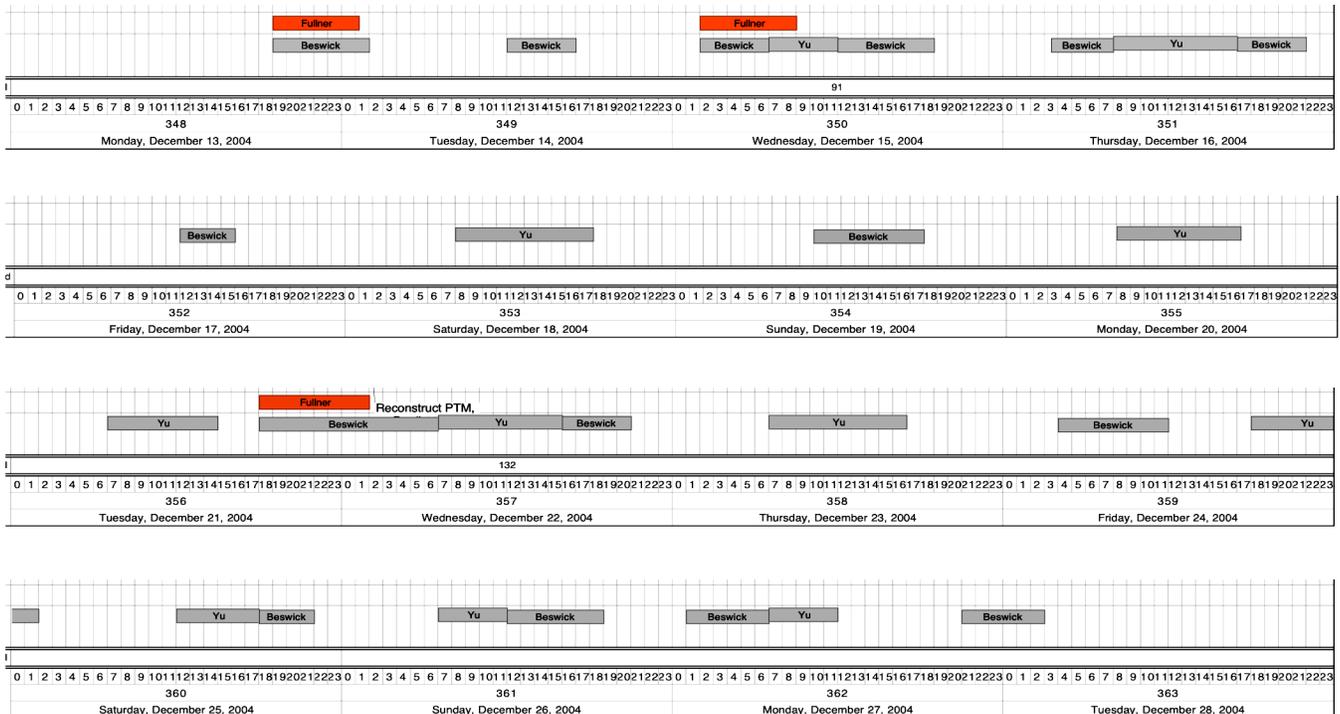


Figure 5. Flight Operations Scheduling For Saturn Tour

At first glance this schedule seems to be a very difficult proposition to support. However, a few key paradigms were put in place that helped ease this busy period:

- Every six days, a system administrator was given a “day” off (such terms can be understood to be somewhat fungible by the round the clock scheduling necessary).

- System administrators would be kept on the same sleep schedule. Avoiding the “jet-lag” effects of a widely divergent sleep schedule would help a great deal to improve the tolerability of a difficult schedule. (As seen above, this could induce interesting schedule divisions.)
- Wherever possible, shifts would be kept to under ten hours in duration.

These approaches, along with some flexibility on the part of the systems administrators, helped a small support staff provide coverage for hundreds of critical shifts over the course of the prime mission, often at highly unusual times and days and dates.

V. Observations and Lessons Learned

In such an effort, a number of caveats, miscellaneous points of wisdom and foundational ideas will emerge. Although not seminal considerations, some observations may prove useful in the design of future systems of this scale. Many of these are well known System Administrator proverbs. As with all such injunctions, your mileage may vary.

1. *“Disks are cheap.”*
Even in 1996 the cost of disk storage was much, much cheaper than the System Administrator and Navigation Engineer time necessary to work around insufficient file system space, or recover a file from a backup tape that could have had an online backup. Working through this problem proved to be the single greatest improvement to system reliability accomplished throughout the mission.
2. *“Buy as much disk as you can afford.”*
After reviewing the painstakingly gathered capacity requirements for disk storage, the lead author (who purposely put in a large margin for each of the Navigation subsystems) is struck by how wildly inaccurate the estimates for file system actually were. In some cases file system usage is off by twenty times the estimated value in 2002. This is not meant however as an accusation of waste or inefficiency, for new techniques and software sets have come into use that were difficult to foresee as part of the estimate made for the functional requirements. It would have been more cost effective to have simply skipped that part of the requirements gathering and used the money saved to buy more disk. (It is also somewhat disturbing to consider that the Cassini Navigation team managed to cross most of the solar system using less than 36 GB of disk for the *entire* Navigation team, where now one Navigation subgroup has problems fitting into 2 TB of disk storage for its day-to-day-operations.)
3. *“Buy the best disk you can afford.”*
SCSI and SAS drives still provide a much greater performance and lifespan than commodity SATA (or older disks). RAID arrays further improve performance and reliability, and further differentiate on such metrics, as well as support options, by cost. As noted, it is sometimes ok to put all of one’s eggs in one basket, but it is reckless to not make the appropriate investments in that basket. This author cannot count the number of times he discovered a disk had failed in the primary disk array by coming into work and seeing a package containing a replacement disk sitting on his chair. This is a vast improvement over frantic phone calls made by Navigation Engineers at 2 am.
4. *“Make your life as easy as possible in the event of a crisis.”*
Complicated failover and backup disaster recovery procedures will not be of help during a crisis when the pressure is on and users are screaming. Recovery procedures should be simple, must be tested, and must work as expected.
5. *“Never delete anything as a System Administrator, for that is what users do... .”*
This does go hand in hand with the first three points. The lead author has become so enamored of version control that he keeps multiple versions of software, toolsets and configurations around. The time saved in not having to recreate a prior configuration for numerous classes of questions and problems is almost magical.

(Besides, the System Administrator area on my servers is the only area that is using remotely close to the initial functional requirement estimates for file system storage!) Furthermore, if a user asks to have something deleted it is likely that a restoration from a backup is soon to follow. This request is often due to a permission problem, or because they do not want to take responsibility for deleting files owned by other users. Instead of possibly creating a bigger problem, find out *why* they are asking you to do their dirty work. It may prove instructive for both you and the user.

6. *“Repair failures consistently and regularly.”*

This is an issue that can occur in systems with multiple redundant functional units and busy System Administration staff. Although the system can tolerate numerous failures before problems lead to work stoppages, it is important that the failure of individual components - no matter how low a priority of the component - be repaired in a regular manner. Otherwise one can find oneself spending a whole afternoon running around fixing trivial printing problems when one discovers that the redundant spare printers were down for repairs as well.

7. *“Have an off-site Disaster Recovery plan.”*

The seventh point is an issue that has been a concern since the beginning of the Cassini Mission. As mentioned above, during the Launch phase of the mission, an extensive, and expensive Disaster Recovery plan was implemented that had an off site Emergency Operations Center located at the Goldstone Communications complex. After launch, support of that effort was terminated due to lack of funding. For a true Disaster Recovery capability, there is no substitute for such an off-site capability. As always budget constraints serve as a practical limit to the design of fault tolerant systems, for, just as not all children get to grow up to be astronauts, as noted above, not all computer systems can be designed to have the fault tolerant capability and reliability of the Boeing 777 Fight Avionics system. This is an understood part of risk management. However, such DR capability (to be able to tolerate a site-wide failure) is an important consideration for fault tolerant and reliability design. Although considering the late phase of the primary mission it is very unlikely that such a capability will be restored, it is *strongly* recommended that future efforts of this size consider such a Disaster Recovery capability.

8. *“Sometimes it is ok to rely on someone else.”*

Although there were some concerns with our partnership with the Flight Operations Network Engineering organization, that partnership has provided a fault tolerant, high speed network for many years now. Although this author continues to maintain that, with adequate funding, such services could be better provided in house, it is clear that the Cassini Navigation computational environment has greatly benefited from the services provided by our peers.

9. *“In Flight Operations there is no such word as ‘surrender’...”*

Finally, the importance of the right attitude of the System Administration staff can not be overemphasized. The stubborn refusal to accept failure is vital for people who serve in this role. This serves as a reminder of both the stress and the reason why people want to take up the compelling challenge of Flight Operations work. Much depends on the right focus of the people who serve to keep these critical systems running that fly the spacecraft. They must be willing to do whatever is necessary, by any means necessary, as former NASA Flight Director Chris Kraft once defined, to “...take any action necessary for mission success.”⁴¹ Flight Director Peter Frank codified what this attitude entails, “To always be aware that suddenly and unexpectedly we may find ourselves in a role where our performance has ultimate consequences. To recognize that the greatest error is not to have tried and failed, but that in trying, we did not give it our best effort.”⁴²

VI. Conclusion

We have described the efforts to maintain, improve and formally overhaul the Cassini Navigation Ground Data System. We have considered such efforts over the history of the mission, from the initial efforts to derive a launch-ready configuration, through a system of gradual improvements to the Navigation computational environment over the Cruise portion of the mission, to a culmination in the formal specification, design and implementation undertaken to support Saturn Tour Operations. As a part of this effort a number of design constraints were

considered, some derived from Navigation Orbital Tour requirements on system performance and operation, and some representing general system design paradigms considered as a result of lessons learned during Cruise. It is hoped that such efforts will find utility with other organizations and missions that may be facing similar computational challenges.

Appendix

As a part of this discussion, after considering the operational constraints and system requirements that have been evaluated, it may be of some interest to examine an overview of the current configuration of the Cassini Navigation Ground Data system. Figure 6 describes the general layout of the Navigation computer environment. While not all of these systems were purchased as part of the initial overhaul of the Navigation computational environment described above, they were purchased as part of a gradual program of hardware upgrades over the course of the mission.

CURRENT CONFIGURATION:

Red-Hat Linux PC Workstations and Servers:

- 29 x High End PC-DELL, P4, 3.06-3.80GHz, 1.5GB RAM.
- 4 x Low End PC-DELL, P4, 2.66GHz, 1GB RAM.
- 3 x PC Server-DELL, dual Xeon, 2.8GHz, 3GB RAM.
- 1 x PC Server-DELL, dual Xeon, 3.7GHz, 2GB RAM.

Sun Solaris Workstations (AMMOS/SFOC Software Support):

- 7 x Sun Ultra 10, 440 MHz UltraSparc II, 1 GB RAM.

Other Server Hardware:

- Network Appliance FAS 3020 8TB (mirrored) Network Disk Array

Cassini Navigation Operations Network Backbone:

- 1000 BT (copper) Ethernet connectivity, 36 GB/sec total.

How Cassini Hardware addresses Navigation Functional Requirements:

- All Linux systems have the performance to handle all Navigation tasks
- but not all Flight software has been ported. (allowed in FRD)
- Estimates indicate that performance close to ten times (10x) Cruise.
- Classic “Star” configuration, with multiple “hot” redundant systems and full mirroring of all Server data, as well as cloned configurations provide highly redundant and available environment.
- 8TB mirrored NetApp FAS 3020 disk array, 1000BT (Gigabit) networking to each system, provide capacity required for Tour.
- Archival: A DVD/CD writing capability is available (8.5 GB max capacity) as well as unlimited tape archival through EOM.



CASSINI NAVIGATION SYSTEM DIAGRAM

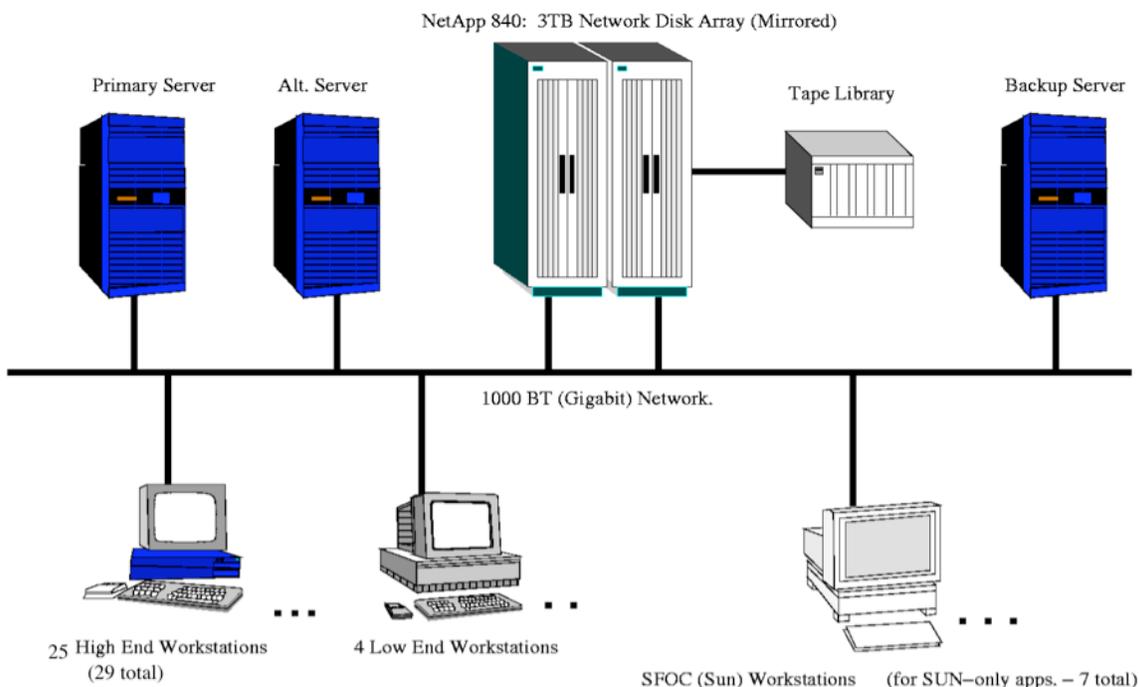


Figure 6. Overview of Cassini Navigation Ground Data System

Acknowledgments

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference to any specific commercial product, process, or service by trade name, trademark, manufacturer or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

¹ P.G. Antreasian et al., "Orbit Determination Processes for the Navigation of the Cassini-Huygens Mission," SpaceOps Conference, Heidelberg, Germany, May 12-16 2008.

² P.N. Williams et al., "Maneuver Operations for the Cassini-Huygens Mission," SpaceOps Conference, Heidelberg, Germany, May 12-16 2008.

³ The Grateful Dead, "Truckin", *American Beauty*, Warner Brothers Records, November 1, 1970.

⁴ "Cassini Interplanetary Trajectory" URL: <http://saturn.jpl.nasa.gov/multimedia/images/mission/IMG000776-br500.jpg> [cited 26 March 2008]

⁵ Wikipedia, "Whack-A-Mole" [online encyclopedia] URL: <http://en.wikipedia.org/wiki/Whac-A-Mole> [cited 18 March 2008].

⁶ Morris, James H. , Satyanarayanan, Mahadev, Conner, Michael H., Howard, John H., Rosenthal, David S. H., and Smith, F. Donelson, “Andrew: A Distributed Personal Computing Environment”, *Communications of the ACM*, Vol. 29, No. 3 March 1986, pp. 188-191

⁷ Coulouris, George, Dollimore, Jean, and Kindberg, Tim, *Distributed Systems, Concepts and Design 4th Ed.*, Addison-Wesley, New York, 2005, p. 519.

⁸ Kranz, Gene, *Failure is not an option: mission control from Mercury to Apollo 13 and beyond*, Berkley Publishing Group, New York, 2003, p. 307.

⁹ “rsync” URL:<http://www.samba.org/rsync> [cited 24 March 2008]

¹⁰ Cheswick, William R. and Bellovin, Steven M., *Firewalls and Internet Security, Repelling the Wily Hacker*, Addison-Wesley, Reading, MA, 1994, pp. 49-131.

¹¹ Chapman, Brent D. and Zwicky, Elizabeth P., *Building Internet Firewalls*, O’Reilly & Associates Inc. , Cambridge, 1995 pp. 55-375.

¹² “Ignite-UX Home” URL: <http://docs.hp.com/en/IUX> [cited 24 March 2008].

¹³ Beswick, R. M., “How to Build and Secure a General Purpose ‘Internet Ready’ Workstation”, *The SANS Institute*, URL: http://www.giac.org/certified_professionals/practicals/gcux/89.php [cited 20 March 2008].

¹⁴ Cheswick, William R., Bellovin, Steven M., and Rubin, Aviel D., *Firewalls and Internet Security, 2nd Ed.*, Addison-Wesley, New York, 2003, pp. 4, 10-14.

¹⁵ Moore, Gordon E., “Cramming more components onto integrated circuits”, *Electronics*, Vol. 38, No. 8, 19 April 1965, pp. 114-117.

¹⁶ Intel, “Excerpts from A Conversation with Gordon Moore: Moore’s Law”, [Video Transcript], 2005 Intel Corporation, URL: ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf [cited 30 March 2008].

¹⁷ Walter, Chip, “Kryder’s law”, *Scientific American*, August 2005, pp. 32-33.

¹⁸ Beswick, R. M., “Cassini Navigation Hardware Requirements”, JPL IOM 312.D/007-2002, Jet Propulsion Laboratory/NASA, Pasadena, CA 30 September 2002.

¹⁹ “SPEC CPU2000 V1.3”, Standard Performance Evaluation Corporation, 7 June 2007, URL: <http://www.spec.org/cpu/> [cited 30 March 2008]

²⁰ Beswick, R. M., “Initial Product Evaluation for Cassini Navigation Upgrades”, JPL IOM 312.D/008-2002, Jet Propulsion Laboratory/NASA, Pasadena, CA 24 November 2002.

²¹ Wikipedia, “Peer to Peer Network” [online encyclopedia] URL: <http://upload.wikimedia.org/wikipedia/commons/thumb/3/3f/P2P-network.svg/150px-P2P-network.svg.png>

²² Wikipedia, “Server based Network” [online encyclopedia] URL: <http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Server-based-network.svg/150px-Server-based-network.svg.png>

²³ “Main Page – SystemImager” URL: <http://wiki.systemimager.org> [cited 24 March 2008].

²⁴ Network Appliance, “Network Appliance Snapshot Technology”, 2004 Network Appliance Incorporated, URL: <http://media.netapp.com/documents/snapshot.pdf> [cited 30 March 2008].

- ²⁵ Gray, Jim and Siewiorek, Daniel P., “High-Availability Computer systems”, IEEE Computer Society, Los Alamitos, CA, September 1991, p. 42.
- ²⁶ Cheswick, William R., Bellovin, Steven M., and Rubin, Aviel D., *Firewalls and Internet Security, 2nd Ed.*, Addison-Wesley, New York, 2003, p. 279.
- ²⁷ Dal Cin, Mario, Meadows, Catherine, Sanders, William H. (eds). *Dependable Computing for Critical Applications 6*, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 249-270.
- ²⁸ Iyer, Ravishankar K., Tsai, Timothy K., and Jewitt, Doug, “An Approach towards Benchmarking of Fault-Tolerant Commercial Systems”, IEEE Proceedings of FCTS-26, 1996, pp. 314-323.
- ²⁹ Iyer, Ravishankar K., Morganti, Michele, Fuchs, W. Kent, Gligor, Virgil (eds). *Dependable Computing for Critical Applications 5*, IEEE Computer Society, Los Alamitos, CA, 1998, pp. 3-18.
- ³⁰ Cheswick, Bellovin, Rubin, p. 259.
- ³¹ Bishop, Matt, *Computer Security, Art and Science*, Addison-Wesley, New York, 2003, pp. 3-6.
- ³² Libes, D. “RFC 1178 – Choosing a name for your computer” August 1990.
URL: <http://www.faqs.org/rfcs/rfc1178.html> [cited 27 March 2008].
- ³³ “Tripwire” URL: <http://www.tripwire.com> [cited 24 March 2008].
- ³⁴ Cheswick, Bellovin, Rubin, pp. 259-277.
- ³⁵ Cheswick, Bellovin, Rubin, p. 260.
- ³⁶ ISS (Internet Security Systems), URL: <http://www.iss.net/> [cited 24 March 2008].
- ³⁷ Skoudis, Ed., with Liston, Tom, *Counter Hack Reloaded, A Step-by-Step Guide to Computer Attacks and Effective Defenses*, Prentice Hall, New York, 2006, pp. 310-319.
- ³⁸ NESSUS, the Network Vulnerability Scanner, URL: <http://www.nessus.org/> [cited 24 March 2008].
- ³⁹ The Center for Internet Security, URL: <http://www.cisecurity.org> [cited 25 March 2008]
- ⁴⁰ “OI’ Lonely” [the Maytag repairman] URL: <http://en.wikipedia.org/wiki/Maytag> [cited 25 March 2008]
- ⁴¹ Kranz, Gene, p. 123.
- ⁴² Kranz, Gene, p. 393.

Copyright 2008 © California Institute of Technology. Government sponsorship acknowledged.